Fault Tolerant Control Systems

a Development Method and Real-Life Case Study

Bøgh, S.A.

Publication date:
1997

Document Version
Også kaldet Forlagets PDF

Fault Tolerant Control Systems - a Development Method and Real-Life Case Study

Ph.D. Thesis

Søren Abildsten Bøgh

Department of Control Engineering
Aalborg University
Fredrik Bajers Vej 7, DK-9220 Aalborg Ø, Denmark.

ISBN 87-90664-01-9
Doc. no. D-97-4198
December 1997

This thesis was typeset using LaTeX2$_\varepsilon$ in `report` document class.

Drawings were made in CORELDRAW™ from Corel Corporation.
Graphs were generated in MATLAB™ from The MathWorks Inc.
The Array Inference Toolbox AIT™ and VISUALSTATE™ from Beologic A/S were used to analyse rule bases and state-event machines.

Preface and Acknowledgements

This thesis is submitted in partial fulfillment of the requirements for the Doctor of Philosophy at the Department of Control Engineering, Aalborg University, Denmark. The work has been carried out in the period from July 1991 to December 1997 under the supervision of Professor Mogens Blanke.

The thesis considers the design of fault tolerant control systems for ordinary industrial processes that are not categorized as high risk applications, but where high availability is desirable. The results presented in the thesis are based on experience from two involvements. A three year project under the Danish Research Council (STVF) with the title "Reliable Control Systems - Systematic Methods for Fault Handling Design", and three and a half years in the development of the attitude control system for the Danish Ørsted satellite. The STVF project contributed with basic and applied research and the satellite involvement gave invaluable experience from a genuine application where mission critical elements were involved.

I am gratefully indebted to my supervisor Professor Mogens Blanke for his guidance throughout the project and also for giving me the opportunity to participate in the Ørsted project.

I also want to thank Rikke Bille Jørgensen who joined me in the STVF project and was a great support throughout these three years.

A sincere thank goes to Professor Ron Patton from Hull University, England, for interesting discussions during my half year stay at York University in 1993.

Furthermore, I greatly acknowledge the assistance from my colleagues within the research group for fault tolerant control and the Ørsted group: Roozbeh Izadi Zamanabadi, Claus Thybo, Rafał Wiśniewski, and Thomas Bak.

Finally, I want to acknowledge the financial support from STVF under grants no. 16-4979 and no. 9500765 and from the Ørsted satellite project.

December 1997, Aalborg, Denmark
Søren Abildsten Bøgh

Summary

This thesis considered the development of fault tolerant control systems. The focus was on the category of automated processes that do not necessarily comprise a high number of identical sensors and actuators to maintain safe operation, but still have a potential for improving immunity to component failures. It is often feasible to increase availability for these control loops by designing the control system to perform on-line detection and reconfiguration in case of faults before the safety system makes a close-down of the process.

A general development methodology is given in the thesis that carried the control system designer through the steps necessary to consider fault handling in an early design phase. It was shown how an existing control loop with interface to the plant wide control system could be extended with three additional modules to obtain fault tolerance: Fault detection and isolation, remedial action decision, and reconfiguration. The integration of these modules in software were considered.

The general methodology covered the analysis, design, and implementation of fault tolerant control systems on an overall level. Two detailed studies were presented, one on fault detection and isolation design and one on design of the decision logic. Two application case studies were used to emphasize practical aspects of both the development methodology and the detailed studies. One was an electro-mechanical actuator in a position control loop for a diesel engine speed governor where the purpose was to avoid a total close-down in case of the most likely faults. The second was a fault tolerant attitude control system for a micro satellite where the operation of the system is mission critical. The purpose was to avoid hazardous effects from faults and maintain operation if possible.

A method was introduced that, after a systematic examination of possible component failures, enables analysis of the relationship between failures and their consequences for the system's operation. This fault propagation analysis is based on coarse models of the subsystems describing the reaction to faults, as for example a variable being zero, low or high. Examples were given that illustrate how such models can be established by simple means, and yet provide important information when combined into a complete system. A special achievement was a method to determine how control loops behave in case of faults. This is not straight forward as the system behaviour depends on the character of

the feedback.

One of the detailed studies were the design of the decision logic in fault handling, realized as state-event machines. Guidelines for the design were provided, based on experience from the two case studies. Methods for verifying correct operation of the decision logic were described, where a completeness check against the fault propagation analysis is able to guarantee coverage of all considered faults.

The usage of software tools to support the development process was illustrated with an off-the-shelf product for constraint logic solving and state-event machine analysis. The coarse system models and the decision logic were analyzed with the tool-box and it was shown how an easy analysis could be performed to verify correctness and completeness of the fault handling design. Experience from this study highlights requirements for a dedicated software environment for fault tolerant control systems design.

The second detailed study addressed the detection of a fault event and determination of the failed component. A variety of algorithms were compared, based on two fault scenarios in the speed governor actuator setup. One was a position sensor fault and the second was an actuator current fault. The sensor fault detection was trivial, whereas the actuator fault was more challenging. The study demonstrated that many existing methods have a potential to detect and isolate the two faults, but also that the research field still misses a systematic approach to handle realistic problems such as low sampling rate and nonlinear characteristics of the system. The thesis contributed with methods to detect both faults and specifically with a novel algorithm for the actuator fault detection that is superior in terms of performance and complexity to the other algorithms in the comparative study.

Synopsis

Denne Ph.D.-afhandling omhandlede udviklingen af fejl-tolerante kontrolsystemer. Der blev fokuseret på den type af automatiserede processer, der ikke har en stor mændge identiske sensorer og aktuatorer til at opretholde sikker operation, men hvor det alligevel er ønskeligt og muligt at opretholde driften i tilfælde af simple fejl. Afhandlingen illustrerede muligheden for at integrere fejldetektion og aktiv omkonfigurering i designet af kontrolsystemet, så automatisk nedlukning fra sikkerhedssystemet undgås.

En generel udviklingsmetode er beskrevet i afhandlingen, som vejledning til kontrolsystem-designeren i de forskellige discipliner indenfor fejltolerant kontrol. Formålet var at stille et værktøj til rådighed, som gør det muligt at overveje fejlhåndtering på et tidligere stadie i udviklingsprocessen end hvad er praksis i dag. Det blev vist, hvorledes et eksisterende kontrolsystem, med grænseflade til et overordnet proces-kontrolsystem, kunne gøres fejltolerant ved at inkludere følgende tre moduler: Fejldetektion og -isolering, handlings-beslutning samt omkonfigurering. Aspekter i forbindelse med implementering i software blev også behandlet.

De generelle overvejelser i udviklingsmetoden om analyse, design og implementering blev suppleret med to detail-studier. Det ene omhandlede algoritmer til fejldetektion og -isolering, det andet design af beslutnings-logikken for fejlhåndtering. To eksempel-studier blev anvendt til at illustrere praktiske aspekter i forbindelse med både den generelle udviklingsmetode og detail-studierne. Det ene var en elektrisk-mekanisk positionsregulering, anvendt til hastighedsstyring på store dieselmotorer, som blev analyseret og designet til at modstå nedlukning som følge af de mest sandsynlige fejl. Det andet var fejltolerant retningsstyring til en mikrosatellit, hvor missionens success er afhængig af korrekt funktion af kontrolsystemet. Formålet var at undgå uheldige følgevirkninger fra fejl, og opretholde aktiv kontrol så vidt det er muligt.

En metode blev introduceret som, efter en systematisk undersøgelse af potentielle fejlmuligheder, kan bruges til at analysere sammenhængen mellem fejlene og deres konsekvens for driften. Denne undersøgelse af fejlenes udbredelse igennem systemet baserer sig på delsystem-modeller med grov kvantisering, hvor udfaldsrummet for en variabel for eksempel bliver beskrevet med nul, lav eller høj. Der blev givet eksempler på, hvordan sådanne delsystem-modeller relativt simpelt kan udvikles og kombineres til en komplet systembeskrivelse, som kan anvendes til en værdifuld undersøgelse af syste-

met. Et specielt bidrag i afhandlingen er en metode til at undersøge udbredelsen af fejl i tilbagekoblede systemer. Systemets opførsel er ikke altid umiddelbar gennemskuelig, da reaktionen afhænger af tilbagekoblingens struktur.

Et af detail-studierne omhandlede design af beslutningsdelen for fejlhåndtering. En realisation med tilstandsmaskiner blev præsenteret og en række retningslinier for designet blev givet udfra praktiske aspekter i forbindelse med de to eksempelstudier. Der blev anvist metoder til at verificere korrekt funktion af beslutningsdelen, idet det er muligt at sikre fuldstændig dækning af de fejl, som analysen omfatter.

Anvendelsen af EDB-værktøjer til understøttelse af udviklingsprocessen blev illustreret med et kommercielt produkt til analyse af logiske kredsløb og tilstandsmaskiner. Det blev vist, hvordan ovenstående komplethedsanalyse nemt og elegant kan gennemføres, når værktøjet benyttes til at analysere de groft kvantiserede modeller sammen med beslutnings-logikken. Erfaringerne fra dette studie viste, hvilke krav der stilles til programmel-omgivelser for fejltolerant kontrol-design.

Det andet detail-studie omhandlede detektion af fejl-hændelser og lokalisering af den fejlbehæftede komponent. En række forskellige algoritmer blev sammenlignet baseret på deres evne til at opfange to specifikke fejlsituationer i positionsreguleringen for dieselmotoren. Det ene tilfælde var en positionsmåler-fejl og det andet var en strømfejl i aktuatoren. Sensorfejlen var triviel, mens aktuator-fejlen var en større udfordring. Sammenlignings-studiet viste, at mange metoder har et godt potentiale for detektion og isolation af de omtalte fejltyper, men også at forskningsfeltet savner en systematisk angrebsvinkel til at håndtere realistiske problemer, såsom lav samplingshastighed og ulinearitetер i systemet. Afhandling bidrog med metoder til detektion af begge fejltyper og specielt med en elegant algoritme til detektion af aktuator-fejlen, som overgår de andre metoder i sammenlignings-studiet i form af præstationer og kompleksitet.

Contents

List of Figures

List of Tables

Nomenclature

Symbols

$\mathbf{x}$	State-space description's state-vector $[i_2\ n_m\ s_o]^T$
u	Input n_{ref}
$\mathbf{y}$	Output vector $[n'_{m\nu}\ s'_{o\nu}]^T$
d	Disturbance Q_l
f_a	Actuator fault Δi_m
f_s	Sensor fault Δs_o
ν	Sensor noise vector $[\nu_{nm}\ \nu_{so}]^T$
$\mathbf{A}_c\ \mathbf{B}_c\ \mathbf{E}_c\ \mathbf{F}_{ac}$	Continuous-time state-space state equation matrices
$\mathbf{A}\ \mathbf{B}\ \mathbf{E}\ \mathbf{F}_a$	Discrete-time state-space state equation matrices
$\mathbf{C}\ \mathbf{F}_s$	State-space output equation matrices
$\mathbf{H}$	System's output transfer function matrix
$\mathbf{G}$	System's input transfer function matrix
$\mathbf{W}$	Transformation matrix for parity space FDI-design
K_v	Benchmark actuator velocity controller gain
T_v	Benchmark actuator velocity controller integral time
I_{peak}	Benchmark actuator power drive peak current limit
I_{rms}	Benchmark actuator power drive mean current limit
f_{tot}	Benchmark actuator total friction referred to servo motor
I_{tot}	Benchmark actuator total inertia referred to servo motor
I_l	Benchmark load inertia of spring-mass model of load, referred to gear output
K_q	Benchmark actuator torque constant for servo motor
N	Benchmark actuator gear ratio
α_s	Benchmark actuator measurement scaling factor
η	Benchmark actuator gear efficiency
K_{nm}	Benchmark actuator noise gain on velocity measurement
K_{so}	Benchmark actuator noise gain on position measurement
i'_m	Benchmark actuator final current in motor including current limits and current faults
i_m	Benchmark actuator motor current if no faults in power drive. After current limits

i_{m0}	Benchmark actuator requested current from velocity controller without current limits
i_2	Benchmark actuator velocity controller integral variable
n_m	Benchmark actuator shaft speed of servo motor
$n_{m\nu}$	Benchmark actuator measurement of servo motor shaft speed including noise
n_{ref}	Benchmark actuator shaft speed reference
Q_{lm}	Benchmark actuator load torque referred to servo motor
Q_m	Benchmark actuator torque from servo motor
Q_{fm}	Benchmark actuator friction torque when load is separated in a spring-mass model
Q_{fl}	Benchmark load friction torque when load is separated in a spring-mass model
Q_s	Benchmark load torque when load is separated in a spring-mass model
s_o	Benchmark actuator output arm position
$s'_{o\nu}$	Benchmark actuator measurement of output arm position including noise and faults
Δi_m	Benchmark actuator current fault as additive input
Δs_o	Benchmark actuator position measurement fault as additive input
r_s	Benchmark residual for sensor (position) fault detection
r_a	Benchmark residual for actuator (current) fault detection
L	Benchmark gain in position measurement fault FDI observer

Abbreviations

ACS	Attitude control system
ADC	Analog-digital converter
AI	Artificial intelligence
AIT	Array inference tool-box from Beologic™
CASE	Computer aided software engineering
CCM	Coil current measurement
CD	Coil driver
CDC	Control distribution concept
CDH	Command and data handler
CPD	Charged particle detector
CSC	Compact spherical coil (vector magnetometer)
CUSUM	Cumulative sum
DC	Direct current
DES	Discrete event system
DoD	Department of Defense, US
EKF	Extended Kalman filter
EMC	Electro-magnetic compatibility
ESA	European Space Agency
FDI	Fault detection and isolation
FDIR	Fault detection, isolation, and reconfiguration

FMEA	Failure mode and effects analysis
FMECA	Failure mode, effects, and criticality analysis
FPA	Fault propagation analysis
FPG	Fault propagation graph
FSM	Finite state machine
FTC	Fault tolerant control
FTCS	Fault tolerant control system
GLR	Generalized likelihood ratio
GPS	Global positioning system
HazOp	Hazard and operability analysis
IFAC	International Federation of Automatic Control
IRU	Inertial reference unit
IMM	Interacting multiple model
LMI	Linear matrix inequality
MIMO	Multiple input multiple output
MTQ	Magnetorquer
NASA	National Aeronautics and Space Administration
NEAR	Near earth asteroid rendezvous
NMP	New millennium project
NN	Neural network
OSD	One-step diagnosis (neural network)
OVH	Overhauser scalar magnetometer
PCT	Procedural control theory
PHA	Preliminary hazard analysis
PI	Proportional integral
PIM	Pseudo-inverse method
PRBS	Pseudo random binary sequence
RCE	Restricted coulomb energy (neural network)
SCT	Supervisor control theory
SEM	State-event machine
SIM	Star imager
SISO	Single input single output
SPRT	Sequential probability ratio test
SS	Sun sensor
SST	Sun sensor thermistor
STVF	Statens Teknisk Videnskabelige Forskningsråd (Danish Research Council)

Terminology

The terminology used in FTCS has only during the recent years approached an agreement in the published material. The Safeprocess Technical Committee of IFAC has compiled a list of suggested definitions (Isermann and Ballé (1997)) which is generally in coherence with the terminology used throughout this thesis. Additional publications with explanations on terminology used below are Bell (1989), ESA-ECSS (1997), NASA (1993), and DoD (1980).

Active fault tolerant system	A fault tolerant system where faults are explicitly detected and accommodated. Contrary to a passive fault tolerant system.
Analytical redundancy	Use of more than one not necessarily identical ways to determine a variable, where one way uses a mathematical process model in analytical form.
Availability	Probability that a system or equipment will operate satisfactorily and effectively at any point of time.
Criticality	A combined measure of the severity of the consequences of an event and the frequency of the event.
Dead-end	A state or combination of states in state event machines that can only be left with a complete reset of the state event machines.
Decision logic	The functionality that determines which remedial action(s) to execute in case of a reported fault and which alarm(s) shall be generated.
Detector	An algorithm that performs fault detection and isolation.
Effector	An algorithm that executes a remedial action.
End-effect	The consequence of a failure mode on the operation, function, or status of a system on top level.
Fail-safe	The ability to sustain any single point failure and retain full operational capability.
Failure effect	The consequence of a failure mode on the operation, function, or status of an item.
Failure mode	Particular way in which a failure can occur.
Fault detection	Determination of faults present in a system and time of detection.
Fault diagnosis	Determination of kind, size, location and time of detection of a fault. Follows fault detection. Includes fault isolation and identification.
Fault isolation	Determination of kind, location and time of detection of a fault. Follows fault detection.
Fault tolerant system	A system where a fault is accommodated with or without performance degradation, but a single fault does not develop into a failure on subsystem or system level.
Functional structure model	model, where the system is broken down into units that perform sub-tasks.
Hardware redundancy	Use of more than one independent instrument to accomplish a given function.

Hazard class	A categorization of fault effects' severity that classifies how dangerous the consequences are to the system's operation, damage to equipment, and damage to the environment.
Logic model	A model of a system where input and output values are described in terms of intervals (e.g. negative, zero, positive).
Magnetorquer	Electro-magnetic coil in a satellite that generates a magnetic moment used for attitude control.
Passive fault tolerant system	A fault tolerant system where faults are not explicitly detected and accommodated, but the controller is designed to be insensitive to a certain restricted set of faults. Contrary to an active fault tolerant system.
Physical structure model	A model, where the physical components are grouped by location and described by their interconnections.
Qualitative model	A system model describing the behaviour with relations among system variables and parameters in heuristic terms such as causalities or if-then rules.
Quantitative model	A system model describing the behaviour with relations among system variables and parameters in analytical terms such as differential or difference equations.
Reconfiguration condition	The worst-case fault effect that is tolerated before a remedial action is executed that accommodates the fault.
Reliability	Probability of a system to perform a required function under normal conditions and during a given period of time.
Remedial action	A corrective action (reconfiguration or a change in the operation of a system) that prevents a certain fault to propagate into an undesired end-effect.
Safety system	Electronic system that protects local subsystems from permanent damage or damage to environment when potential dangerous events occur.
Severity	A measure on the seriousness of fault effects using verbal characterisation. Severity considers the worst-case damage to equipment, damage to environment, or degradation of a system's operation.
Supervision	Monitoring a physical system and taking appropriate actions to maintain the operation in the case of faults.
Supervisor	A system that performs supervision by means of fault detection and isolation, determination of remedial actions, and execution a corrective actions.

Chapter 1

Introduction

This thesis considers the design of fault tolerant control systems (FTCSs) for the category of processes running daily that are not considered high risk, but where increased availability is desired to improve profitability. Methods exist today for design of fail-safe systems, but these are generally inappropriate for ordinary processes due to development expense and high cost of additional hardware. It is usually feasible to extend the abilities of the control system to monitor the consistency of the available information and, in case of anomalies, reconfigure the process to continue operation or make a safe close-down. It is the aim of this thesis to provide a general methodology for the development of FTCSs with guidelines on the individual steps in the development process. The methodology is approached from an engineering point of view, where practical experience from two case studies forms the basis of the investigation. The techniques presented within the thesis are believed to be applicable for industrial use and implementable with the skills of a regular control engineer.

1.1 Background and Motivation

Todays requirements to performance of processes combined with increased complexity of the equipment necessitate a high level of automation, where some of the typical operator tasks are taken over by computers and where monitoring systems provide improved diagnostic information. It is technically feasible to augment a standard control system with fault handling capabilities, but a general methodology to do this is missed. Traditionally, the possibility of failures is taken into consideration at a very late point in the development of process control systems, and the solutions may therefore be very expensive when they necessitate modifications in the design. It is more sensible to pay attention to faults in the early analysis phase. A systematic approach is then needed that gives a consistent design and assures reliability of the augmented control system. It is crucial that the fault handling system is extremely trustworthy, so particular reliable

techniques must be applied during development and implementation.

The motivation for including fault handling in ordinary control systems comes from industrial experience. Many examples exist where simple faults have caused a close-down or even damage to equipment and thereby an expensive production stop. There is a growing demand for on-line fault accommodation in order to prevent such production stops. It is particularly important in feedback control systems that faults are handled rapidly, because the controller may amplify the effect of a fault. If, for example, a position sensor fails zero, this may accelerate the controlled device and lead to dangerous situations or permanent damage further on in a controlled process. The risk of failures is inevitable; it is not a question of *if* it happens, but merely *when* it happens. Failures are likely to occur due to aging and wear, but also as human errors in connection with establishment and maintenance. It is therefore desired that fault handling is included as a standard practice in the development of control systems.

Among the numerous examples on serious failure situations, two cases are drawn as appetizers:

1. The velocity measurement of the conveyer belt in a beer pasteurizer failed zero. The temperature of the water that brings the beers to the pasteurizing temperature is adjusted by the number of beers passing by, so the zero fault caused the temperature to drop. As a result 65000 beers were maltreated before the failure was discovered. (Frydkjær (1997))

2. The Ariane 5 rocket exploded on June 4th 1996 thirty-seven seconds after lift-off. The reason was a software exception in the inertial reference unit (IRU) that provides attitude and trajectory information for the control system. The exception caused the normal attitude information to be replaced by some diagnostic information that the control system was not designed to understand. As the control system interpreted the data as normal attitude information it reacted by turning the thrusters to their limit with the consequence that the rocket turned over and the self-destruction system made an end to the $8.5 billion program (see report from the inquiry board (Lions (1996)) and a press release in Space News (de Selding (1996))).

The severe consequences of these failure cases could have been avoided or at least mitigated. In the first case a simple consistency check on the velocity measurement could have been used to detect the fault and give an alarm and possibly continue operation with an estimated velocity. In the seconds case, the malfunction of the IRU can be considered as a sensor fault as seen from the control system point of view, although it is actually a software design error that should have been found during testing. A consistency check of the attitude information against the expected attitude could have been used to make a deliberate reaction to the malfunction and possibly saved the mission.

1.2 Overview of the Field

In a broad perspective many of the elements for FTCS design exist today and some of them have matured in real-life systems. What is generally missing is a unified and systematic approach which combines the individual steps and gives a coherent design that is applicable in practice.

Methods for detecting faults and determining their location and characteristics have been studied intensively during the last two decades. A large variety of techniques today are able to extract explicit information about faults by processing measurable signals with dynamic and static models of the known relation between the signals. Quantitative methods are used when analytical models of the processes exist, and qualitative methods are used when only heuristic information is available. The discipline is still on an academic level, although more and more application studies appear. Research in the field has followed different theoretical directions, but the final results show that several techniques are basically equivalent. It is therefore necessary to perform comparative studies that identifies overlap between the techniques and uncover the application areas that the individual methods can be used within. The combined design of fault detectors and controllers has, furthermore, been sparsely considered in the recent years. In this two-sided design problem, robustness to faults are given higher priority with a penalty on controller performance.

The next step in fault handling is to determine what to do when a fault is detected. Methods for alarm generation and presentation of diagnostic information are well established in monitoring systems and widely applied in industry today. It is more problematic with techniques for automatic intervention in the process that seek to decouple the failed component and continue operation. This is very application dependent due to limited sensor and actuator redundancy. For specific processes it may be possible to substitute a failed measurement by an estimate based on related measurements or decouple a failed subsystem entirely, possibly with graceful degradation of the performance. General design techniques are missing today that supports the design engineer to include such facilities during the construction of a process. It is normally feasible to add a few extra input-output channels on a computer or install some extra relays in the wiring, if this is done early in the development process. A systematic approach to this is possible if faults are considered as early as in the analysis phase of the process automation. More advanced techniques with on-line and off-line redesign of control algorithms have been developed within the aviation industry, but these are applicable only for ordinary processes that posses a high level of hardware redundancy which is not typical nor economically favourable.

A survey of state-of-the-art in the elements of FTCS is provided in section 2.3.

1.3 Objectives and Contributions

The aim of this thesis is to provide a coherent development methodology for the design of FTCSs. Guidelines for the individual steps from analysis through design to implementation and test are given and illustrated with examples. It is a particular intention with this work that design engineers are encouraged to incorporate the design for fault tolerance at an early stage in the development process.

Within this framework, the main contributions of the thesis are the following:

- A method for modelling component faults and their propagation based on a *logic* description is presented in section 3.2.1 and section 3.2.2. A matrix technique was presented by Blanke (1996) based on a boolean algebra using AND and OR operators. This technique is further developed in the thesis to cover any combination of logic operations and also work with multi-valued logic variables. A method for analyzing the logic models with a software tool is presented, which makes it feasible for the design engineer to include fault considerations early in the analysis phase and thereby draw the attention to details that need concern. Earlier results on fault modelling and propagation analysis in FTCS design are given in Blanke *et al.* (1993) for the general case and in Bøgh *et al.* (1995) with examples from a satellite case study.

- The problem of *feedback loops* in logic fault propagation models is treated in section 3.2.2. Related studies on this problem can for example be found in Shafaghi *et al.* (1984) and Yang *et al.* (1997) in connection with automatic fault tree generation for reliability assessment. These methods are developed to calculate probabilities for success or failure of a system, but in the scope of this thesis, it is only necessary to consider two issues; first, determination of the effects on system operation from component failures, and secondly, identification of possible oscillation of the real system caused by the feedback loop. The thesis presents a simple and practical solution to these problems that has a potential to be automated in software. The issue of logical feedback was also studied in Jensen *et al.* (1994) and Blanke (1996).

- Implementation of the decision logic for fault handling as *state-event machines* is illustrated in section 3.4.3 and section 5.5 based on earlier work in Blanke *et al.* (1997) and Zamanabadi *et al.* (1996). It is of particular interest that such logic is verified for *correctness* and *completeness*. A novel method for checking completeness of the supervisor's decision logic functionality against the above fault propagation model is presented in section 3.2.8. Additional checks for correctness and consistency of the state-event machines are illustrated in section 5.6. All the verification techniques are developed for practical use and do not make use of theoretical approaches from computer science. Earlier results on completeness check with application to a satellite are available in Bøgh *et al.* (1997).

- Section 3.2 presents a general development methodology for fault analysis of a system, that provides consistent specifications for the design of the FTCS elements. This development methodology has previously been considered with key results available in Bøgh *et al.* (1995), Blanke (1996), Blanke *et al.* (1997), and Bøgh *et al.* (1997).
- A three-level architecture for the implementation of an FTCS is considered in section 3.1. The organization into three levels were originally proposed by the department and recently considered in Blanke *et al.* (1997) for the general case and in Bøgh *et al.* (1997) for application to a satellite. This thesis contributes with detailed application studies of the implementation of FTCS for an electro-mechanical actuator (section 3.4) and a satellite (section 5.7), where problems with implementation in multi-tasking environments are in focus.
- Analytical methods for detection and isolation of faults are considered in chapter 4. A comparative study between a variety of approaches is presented with the purpose to highlight strong and weak sides of advanced methods available in 1997. The study is based on publications from nine research groups that applied their techniques to two specific failure scenarios in a real-life electro-mechanical system. This thesis contributes specifically with simple and efficient algorithms for detection of the two fault types.

A significant contribution of this thesis is the complete design, implementation, and test of an FTCS for a micro satellite described in chapter 5 and previously published in Bøgh *et al.* (1995). The experience from this development project has given invaluable contributions to the typical practical problems encountered in FTCS design.

1.4 Thesis Outline

The thesis is organized as follows:

Chapter 2 places the field of FTCSs in a wider perspective by introducing the related scientific disciplines that must be considered to create a running fault tolerant system. This leads to a formulation of the roles of FTCSs and an outline on how these roles are approached in the thesis. A presentation of state-of-the-art in the topics of FTCS is then given.

Chapter 3 outlines a general development methodology for design of FTCSs. It is first identified that three additional modules must be added to an existing control system for implementation of fault tolerance. A general eight-step recipe is then presented for the analysis of a system leading to design of the three additional modules . The applicability of this recipe is illustrated with the design of an FTCS for an electro-mechanical actuator.

The chapter finally discusses the implementation of this FTCS in combination with an existing controller.

Chapter 4 is devoted to the details of fault detection and isolation algorithms that are the substance of one of the three FTCS modules. Nine different techniques are introduced and results from application to two fault scenarios in the electro-mechanical actuator are presented.

Chapter 5 presents a complete design of the FTCS for the micro satellite application. The design follows the eight-step procedure described in chapter 3 but with emphasis on the design of the fault handling decision logic. The chapter illustrates how the decision logic of a complex supervisor can be realized as a set of state-event machines, how it can be implemented in multitasking software, and how the entire FTCS can be verified and tested.

Chapter 6 finally gives concluding remarks and recommendations for future work.

Chapter 2

Fault Tolerant Control Systems in Perspective

Fault tolerant control systems are considered with the purpose to increase system reliability. A system is reliable if it is able to perform its required function within stated conditions. Increased reliability can be achieved either by ensuring that faults cannot occur (simple but expensive), or by accepting the inevitable risk of faults and design the system to be fault tolerant (cheap but complex). With the purpose to define the role of FTCSs this chapter first discusses the related areas within reliable systems design. This leads to a presentation of the approach taken in this thesis and an outline of the topics considered. State-of-the-Art in these topics is then presented.

2.1 Disciplines Related to Fault Tolerant Control

The design of reliable systems is an interdisciplinary field that covers a wide range of research areas ranging from process instrumentation manufacturing, over software design, to implementation in computers. A system is only reliable if the risk of failures has been considered in each area and subsequently designed anticipating this risk. Reliability and fault tolerance in connection with control systems are considered in the following domains:

Safety systems. Safety systems are systems of logic that detects the occurrence of potential hazardous events and performs a close-down of the process to protect equipment and people. Safety functions are applied on a subsystem level and are independent of process monitoring and control systems. Safety systems may perform unnecessary close-down in some situations where the FTCS would be able to continue safely despite the failure event, because it applies information on a higher level to derive decisions.

In the development of FTCSs this means that some safety functions must be substituted by FTCS actions, which demands the FTCS design to be extremely reliable (Hignett (1996); Balls *et al.* (1988))

Reliability of electrical and mechanical equipment. The ultimate goal is that instruments used for control never fail. This is unrealistic and often unattainable due to factors like cost and weight, so the risk of failure must be accepted. Risk analysts apply *reliability theory* to calculate the likelihood of successful operation given a specified failure rate of each component. This field of research emerged with the increasing requirements to safety of high risk systems as nuclear plants and aircrafts, but some of the principles can be used to enhance availability and profitability of the million of processes running daily that are not considered as high risk. A formal method for the assessment of faults, developed within reliability theory, is *failure mode and effects analysis* (FMEA). Basic principles from this method are used within this thesis. Data banks with failure probabilities of industrial used components have matured over the years, and this information (if accessible) should be used in the assessment of which faults to consider in the FTCS design. A practical guide to reliability assessment is provided in Hignett (1996) and an elaborate introduction to reliability theory is given in Kapur (1977).

Reliability of computers. Error-free performance of computer hardware is a prerequisite for the FTC methods considered in this thesis. *Fault tolerant computers* have been developed using hardware and software techniques like parallel processors, watchdogs, parity checking, built-in error recovery, and dedicated test programs (see e.g. Nelson (1990) and Avizienis (1997)). These protective functions work on a local hardware level and are therefore inherently different from the objective of FTC on a high level, but diagnostic information shall be made available for the FTCS.

Reliability of software. Correct performance of software is the topic in *fault tolerant computing*, where software design and implementational architectures are of interest (see for example Holding (1991)). An introduction to the basic concepts of computing systems dependability is provided in Laprie (1987).

Distributed control systems. Control systems implemented in distributed architectures are vulnerable to transmission faults, execution errors of application tasks, and timing errors on the distributed information. It also opens a prospect for dedicated supervision systems that monitors parallel applications and reconfigures the operation of the network. Design of fault tolerant real-time protocols is a key subject in the field of *dependable real-time systems*. On-line monitoring for error detection in distributed process control systems is treated in Drejer (1994).

Intelligent control. There has been a wide research interest in extending traditional simple feedback loops (e.g. PID) to include additional autonomous functions that sup-

port the installation process and improve the on-line control performance. The control group in Lund, Sweden, uses the term *intelligent control* to cover functions like automatic tuning (Åström and Hägglund (1995)), adaptive control (Hägglund and Åström (1997)), gain scheduling, fault diagnosis, automatic process analysis for controller assessment and design (Åström (1991*a*)), and on-line process monitoring (Hägglund (1995)). A survey of techniques for intelligent control is provided in Åström (1991*b*). Fault tolerant control can be considered as a sub-task within this definition, and many of the techniques developed for FTC can also be applied to the other tasks. Åström (1991*b*) suggests a separation of control algorithms and decision logic in different modules, which is comparable to the approach taken in this thesis. An interesting aspect with this is the interaction between the decision logic and the dynamics of the controlled process. A formal approach to analyzing this problem is taken by Morse (1996) that uses theory from *discrete event systems* (DES) to develop a supervisory scheme for SISO systems to switch between a set of linear set-point controllers when large changes in the operating conditions are explicitly detected.

Passive fault tolerance. Control algorithms are designed to compensate for deviations in the control variable whether it comes from external disturbances, aging sensors and actuators, or failing equipment. In some fault cases a feedback control law attenuates the effect, in other cases it magnifies the effect. Traditional design techniques as robust and adaptive control are able to handle a limited class of faults that has only a small and bounded effect on the system. Explicit status information on individual components is generally not available in these approaches, so it is not guaranteed that the most appropriate action is taken. This is called *passive* approach to fault tolerance. *Active* approaches to fault tolerance differ from this in the way that explicit fault information *is* generated on-line and a proper action can be taken. A further explanation on the difference between passive and active approaches is given in Patton (1997). An example on passive fault tolerance is presented in Veillette *et al.* (1992) that applies the $\mathcal{H}_\infty$-technique to find the set of controllers that provide guaranteed stability in case of sensor and actuator outages in MIMO systems.

Self-validating components. There is an on-going development of intelligent sensors and actuators (Isermann *et al.* (1993), Benítez-Pérez *et al.* (1997), Yang and Clarke (1997)), where more functionalities are incorporated into the device including an integrated fault detection system that provides diagnostic information in parallel with the measurement and control channels. Intelligent components are a natural element in FTC, but as they work on local information only, they do not replace high-level FTCSs.

Fault tolerant control. Active FTCSs are designed to deliver a deliberate reaction in case of faults. Faults in sensors and actuators, anticipated by design, are handled by a supervisory system that detects their occurrence and reconfigures the system to stop the faults from developing into failures at a higher level. Whereas fail-safe systems are

designed to withstand any single failure, FTCSs for ordinary processes is designed to tolerate only the most critical faults. The purpose is not to prevent a shut-down by any means, but to increase availability and avoid hazardous actions in case of simple faults.

2.2 Thesis Approach

In the light of the discussion in the previous section this thesis treats the design of fault tolerance in control systems for ordinary processes that are not required to be fail-safe. Typically, such systems do not have a stated requirement for "the probability of failure" or inversely a "required up-time" as is the case for fail-safe systems, so it is not necessary to formally apply techniques from reliability theory. The requirement is usually to improve *availability* of the process, while keeping the development cost and additional hardware within strict limits. It is then accepted that the process runs a degraded mode in case of faults, instead of investing in additional expensive hardware.

The thesis considers control systems in general, from local subsystem controllers in large plants to high level controllers, but with the common aspect that an increased level of availability is desirable and feasible. Feasibility in this context means that some information about the occurrence of faults shall be available (redundant signals, process knowledge, statistical properties, etc.) which makes fault detection and reconfiguration possible. Such information can sometimes be derived by combining the measurements and knowledge of several subsystems.

The subject of this thesis falls within the area of *active* methods for fault tolerance, where faults are explicitly detected on-line, and the system is reconfigured to accommodate the fault. In this area the content of the thesis can be categorized into the following three topics:

1. a general development methodology for including on-line fault detection and reconfiguration in process control design,
2. design of fault detection, isolation, and reconfiguration algorithms, and
3. design of the decision logic of supervisory systems for fault handling.

The general guidelines presented in the thesis are the outcome of experience from two case studies. A complete FTCS for the attitude control of a micro satellite has been developed and given valuable experience with practical aspects from analysis, design, implementation, and test. The second application is a speed governor for ship diesel engines. A laboratory setup of the equipment, which enables repeated faults scenarios, has been used in the study of real-time fault detection and isolation methods (FDI). This application was suggested as an international benchmark for FDI methods and several research groups have participated with a large variety of approaches. The results are compiled in this thesis with an exceptional opportunity to judge the practical applicability of the various methods.

It is the intent with this thesis to provide practical guidelines with illustrative examples on each step in the development of FTCSs from analysis to implementation and test.

2.3 State-of-the-Art in Fault Tolerant Control Systems Design

This section describes state-of-the-art in FTC with focus on the three topics of this thesis outlined in the previous section. With few exceptions the literature available today focuses on specific details within FTC, so the combined usage of the various disciplines is still an open area for interesting research. An exhaustive overview of the status of FTC in early 1997 is provided in Patton (1997).

2.3.1 General Development Methods

General strategies for development of technical processes have existed for decades and are well-established in industry. One of the most applied approaches is the *water-fall model* shown in figure 2.1.

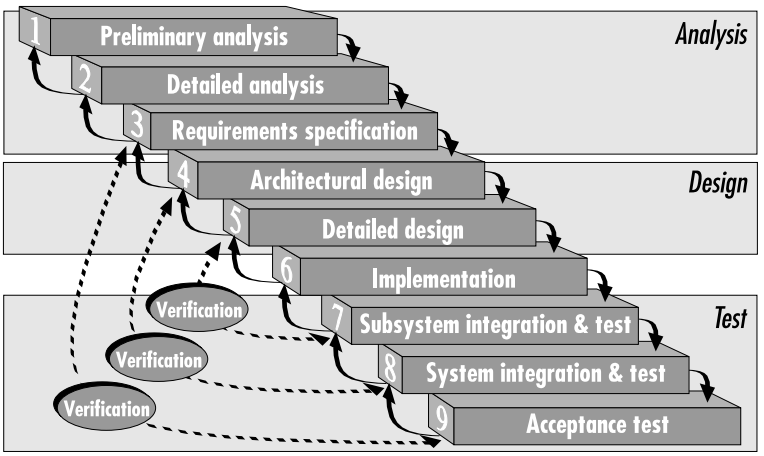


Figure 2.1: *The water-fall model for development of technical processes.*

Detailed guidelines for the development of processes and software in general using the water-fall model (or *V-model* if the boxes are arranged in a V-shape) are provided in Pressman (1988), Biering-Sørensen *et al.* (1990), or Mazza *et al.* (1992). It is more problematic when it comes to specific issues of fault handling. Most publications on fault handling focus on dedicated methods with applications that suit the problem, whereas

only few consider the issue of selecting a proper method for the present problem. Questions that need answers are: In which phase in the development shall the first steps of fault analysis be taken ?, how are functional requirements transformed into specifications for fault tolerance ?, which modelling paradigm shall be used and how accurate shall it be?, what is the best FDIR design method for the present fault handling problem ?, how is fault tolerance adequately implemented along with the control system ?, and what special problems exist in the verification and validation of FTCS ?. Leitch (1993) recognizes some of these questions in connection with diagnostic systems and gives some remarks on model selection and the relationship between requirements and system specification. Design strategies for high availability and safety, based on risk assessment, are briefly discussed in Lauber (1991) with special focus on how to design fault-free software (use of CASE tools) and fault tolerant software (use of redundancy or "diversity"). It is an aim with this thesis to address the above questions and suggest a number of actions in relation to the water-fall model, which promote systematic development of FTCSs.

While general development strategies for FTC are missing, there has been a much wider interest in methods for modelling faults and describing the system behaviour in case of faults. Fault modelling methods, as for example *FMEA* (DoD (1980); Ford/General Motors/Chrysler (1995)) are developed in the field of risk analysis. An overview of these methods are found in Bell (1989) and Jørgensen (1995), and an interesting discussion in the light of robotic fault tolerance is given in Visinsky *et al.* (1994).

The issue of system modelling is different whether it concerns the structure of large complex processes or detailed information for FDI algorithm design. In FDI a mathematical model with sufficient accuracy is needed, and it is usually restricted to only one modelling paradigm. This is different in large scale plants that typically include unlike subsystems described in different levels of detail and with different modelling techniques (Quantitative/qualitative variables, linear/nonlinear models, static/dynamic models, event-driven/time-driven descriptions, discrete-time/continuous-time models, etc.). *Graph theory* based methods for representing the structure of large systems have been developed by the AI community to support diagnosis for supervision and maintenance (e.g. Misra (1994); Cassar *et al.* (1994)). These methods use the causal relationships between subsystems to describe the propagation of faults. Typical methods include *fault tree analysis* and *event tree analysis* (see Hignett (1996) for details). Ideas from the graph-based methods are used in this thesis although formal graph theory is not applied.

Hierarchical decomposition of large complex systems with respect to fault diagnosis is discussed in Misra (1994) and a method is suggested that combines a *physical* structure model and a *functional* structure model. The similar approach is taken in this thesis for design of FTCS.

An interesting concept, that supports FDI design, is presented in Cassar *et al.* (1994) where methods from graph theory are applied to capture the structure of large systems and the behaviour of subsystems in a software tool. The tool produces lists of (possible) detectable faults and non-detectable faults, and is furthermore able to suggest prelimi-

nary FDI algorithms.

2.3.2 Fault Detection and Isolation Algorithms

Algorithms for FDI are required in the detailed design phase of the FTCS development process. The methods are categorized into *signal based* and *model based* techniques. Signal based methods detect faults by testing specific properties of single measurements like mean value, limit value, spectral power, and variance. They have, clearly, a restricted use as they apply to separate signals only. Model based fault detection has a wider range of application, and are normally performed in two steps: *Residual generation* and *residual evaluation*. Residuals are generated by comparing the expected behaviour of the system with the measured behaviour, where the expected behaviour is known from some model of the subsystem. The residuals are normally close to zero and become significantly non-zero in case of faults. Events are then detected by observing changes in the residuals. The pattern of detected events may provide some isolation between more possible faults, but further fault isolation (or diagnosis) can be achieved by evaluating the temporal and spatial characteristic of the pattern against some statistical or probabilistic knowledge.

Two main trajectories have been followed in model based residual generation: *Qualitative* (heuristic) methods and *Quantitative* (analytical) methods. The qualitative methods originate from the above mentioned diagnosis problem of large scale systems whereas the quantitative methods are signal processing units that filter available measurables. Qualitative methods are typically applied for fault isolation and diagnosis but studies on qualitative residual generators are also present (Lunze and Schiller (1997); Calado and Roberts (1997)) typically when the dynamics of the process is not very well known. Qualitative FDI is not considered in this thesis, so the reader is referred to Frank (1996) for an introduction and Lunze and Schiller (1996) for a discussion on the applicability of these methods for FDI in dynamical systems. Verbruggen *et al.* (1995) introduces basic elements of expert systems, outlines their application in fault diagnosis, and compares a number of tools.

Systematic research in quantitative model based residual generation algorithms began with a survey in Willsky (1976) and today mainly three classes of residual generators exist: *Observer based approaches*, *parity space approaches*, and *parameter estimation approaches* (Tzafestas and Watanabe (1990); Frank (1990); Patton and Chen (1996); Frank (1996)). The three methods have different properties but also similarities, which seems logical as they operate on the same input-output signals of the process. The major aspects under concern are the following:

- robustness to modelling uncertainty,
- robustness to unknown inputs (disturbances and noise),
- ability to isolate between more faults,

- ability to detect incipient and/or abrupt faults (frequency response),
- detection of additive and multiplicative faults in linear model descriptions, and
- application of nonlinear methods,

A very large number of techniques have been applied to solve these problems, with different areas of applicability.

The principle in observer based approaches for FDI is to estimate the system outputs with a *Luenberger observer* for the deterministic case or a *Kalman filter* for the stochastic case and then use the estimation errors (or innovations from the Kalman filters) as residuals. The observer gain is chosen to fulfill the requirements for robustness, isolation, and desired frequency response. Various methods for the design of a proper observer gain have been suggested: *Eigenstructure assignment* (Patton and Kangethe (1989); Patton and Chen (1991*b*)), *unknown input observer* (Frank and Wunnenberg (1989); Hou and Müller (1994)), *Kronecker canonical form* (Frank and Wunnenberg (1989)), *fault detection filters* or fault sensitive filters (Beard (1971); Jones (1973); Douglas and Speyer (1996)), and *frequency optimization* based on a factorization of the input-output transfer matrix (Frank and Ding (1994)). Recent developments in the application of Kalman filters are found in Tzafestas and Watanabe (1990); Basseville and Nikiforov (1993); Nikhoukhah (1994); Keller and Darouach (1997). A bank of observers or Kalman filters with distinct properties can be used in parallel to increase the degree of design freedom (Frank and Wunnenberg (1989); Frank (1996)). Different structures of the observer scheme are used to suit the requirements for how many faults shall be detected, how many faults shall be isolated, and how many faults shall be detected and isolated if they occur simultaneously (Frank (1990)).

In the parity space approach residuals are computed as the difference between measured outputs and estimated outputs and their associated derivatives. The primary residuals are shaped by multiplying a transformation matrix $\mathbf{W}(\cdot)$ designed to make the residuals insensitive to unknown inputs and provide isolation between faults. The method handles both additive faults (leads to $\mathbf{W}(s)$, a constant polynomial or rational matrix) and multiplicative faults (leads to $\mathbf{W}(t)$, a time-varying numerical matrix). The parity space approach is described in Gertler (1993, 1997*a*). The parity equations can also be derived from a state-space model of the system (Chow and Willsky (1984)).

The parameter estimation methods for FDI are based on the idea that faults typically effect the physical coefficients of the process. By continuously estimating the parameters of a process model, fault detection residuals are computed as the parameter estimation errors. As the parameters of a model are not always the same as the physical process coefficients the inverse mapping from model parameters to process coefficients must exist and be known, if complete fault isolation is required. Different methods for parameter estimation in FDI have been studied: *Least squares estimation*, *instrumental variable concept*, *output error methods* (Isermann (1984, 1993, 1997)), *sliding mode estimation* (Hermans and Zarrop (1997)), *neural network estimation* (Han and Frank (1997)) and *extended Kalman filters* (Tzafestas and Watanabe (1990); Walker and Huang (1995)).

Most of the research within these three categories considers only linear models of the system. The parameter estimation approach can also handle nonlinear systems if the model is linear in the parameters (Frank (1995)). A survey of nonlinear observer methods and fault diagnosis in bilinear systems is provided in García and Frank (1997).

The three residual generator design approaches are inherently different, but recently some similarities have been studied. It has been shown (Patton and Chen (1991*a*)) that the original parity space equation (where $\mathbf{W}(s)$ is a rational matrix) is a special case of the observer approach, where the observer is designed dead-beat. It is furthermore shown that similar results to the observer methods can be achieved by filtering the parity equation residuals through an adequate filter. Similarities between the parity equations and parameter estimators have been studied by Gertler (1997*b*) and the relationship between observer based methods and parameter estimation has been treated by García and Frank (1996).

The purpose of the residual evaluation part of FDI is to detect when the residuals have changed sufficiently to make a decision credible. The most widely used method is to make a binary decision from a comparison between the residual and a *fixed threshold*.

Increased robustness to modelling uncertainties is achieved with *adaptive thresholds*, where the thresholds are determined on-line depending on the excitation level estimated from measurable inputs. Significant contributions include the threshold selector by Emani-Naeini *et al.* (1988), threshold determination in the frequency domain by Ding and Frank (1991), and an alternative frequency approach based on maximum singular values by Isaksson (1993). Surveys of adaptive threshold techniques are provided in Patton and Chen (1996) and Frank (1996).

A large variety of detection techniques are available in the field of *statistical decision theory* such as generalized likelihood ratio test (GLR), sequential probability ratio test (SPRT), χ^2-test, and cumulative sum test (CUSUM). A comprehensive study of these techniques is available in the book by Basseville and Nikiforov (1993). Surveys are available in Tzafestas and Watanabe (1990), Basseville (1988), and Basseville and Nikiforov (1993).

Additional alternatives to threshold tests and statistical tests are *Fuzzy decision making* (Montmain and Gentil (1993); Frank (1996); Füssel *et al.* (1997); Kiupel and Frank (1997); Lee and Vagners (1997); Miguel *et al.* (1997)) and *pattern recognition* techniques (e.g. *neural networks*) (Himmelblau *et al.* (1991); Kovio (1994); Köppen-Seliger and Frank (1995)).

2.3.3 Reconfiguration Methods

The purpose of reconfiguration is to make the control system insensitive to the effect of a failed component. A particular reconfiguration action can be determined á priori in the design phase or by on-line optimization after a fault occurred. The techniques are thus

categorized into *off-line* methods and *on-line* methods. Some authors confusingly call these categories as *reconfigurable* respectively *restructurable* redesign methods.

A substantial number of publications are found in the field of control of *fly-by-wire* aircraft, where advanced algorithms are designed to fully exploit the high level of redundancy and diversity in sensors and actuators to assure the control system to be fail-safe. Comprehensive surveys of these methods are available in Patton (1997) and Baumgarten (1996). Although several of the methods suggested within aerospace in principle are passive (no explicit FDI information is used), they are included here because they are also applicable in an active scheme. *Control law re-scheduling* is an off-line method, where pre-computed control gains are selected from an estimate of the aircraft impairment status (Moerder *et al.* (1989)). This method is very dependent on correct operation of the FDI system, as incorrect FDI information may lead to serious malfunction. Zheng *et al.* (1997) demonstrates a method to handle these FDI robustness problems by using *linear matrix inequality* (LMI) theory in the synthesis of the feedback control laws. Another off-line approach is the *pseudo-inverse method* (PIM), where the controller gain matrix is re-computed directly by equating the closed-loop transition matrices of the failed and the non-failed systems (see e.g. Gao and Antsaklis (1991)). This calculation involves the estimation of the pseudo-inverse of the failed system's input distribution matrix. Although the method does not guarantee stability, it has been suggested for on-line use by for example Ostroff (1985). An approach related to PIM is the *control distribution concept* (CDC) reported in Celi *et al.* (1996). This method is dedicated to actuator failures only and also needs an estimate of the input matrix of the failed system. It computes a control distribution matrix based on a minimization of the L_2-norm of a quantity that depends on the difference between the failed and non-failed systems and a constraint on the control demands.

The on-line methods apply information about the damaged system to calculate new controller parameters. *Feedback linearization* techniques are suggested, where an adaptive base-line controller is changed on-line by the output from a parameter estimation algorithm. This method is widely used for normal system changes, but is also applied for fault tolerance (Ochi and Kanai (1991)). The passive *model following* approach is also used for FTC in aerospace. Here, the controller gains are computed on-line either by requiring the system trajectories to follow the desired trajectories (explicit model following, see e.g. Morse and Ossman (1990)) or by minimizing a quadratic function of the actual and modelled state rates (implicit model following, see e.g. Huang and Stengel (1990)).

Additional relevant methods for reconfiguration have been studied in other application areas, such as high-speed ships and missile guidance systems (Rauch (1995)), robot control (Visinsky *et al.* (1994)), underwater vehicles control (Payton *et al.* (1992)), and tandem helicopter control (Celi *et al.* (1996)).

2.3.4 Supervisor Decision Logic Design Methods

The third major contribution of this thesis concerns the design of the supervisor's decision logic for fault handling. The task of the decision logic is to select the most suitable action(s) in the controlled process based on information about detected events. The major issues of concern in the design for FTC are:

- How to map the requirements for FTC into design specifications for the decision logic design.
- How to combine fault handling with existing operational command and monitoring systems.
- How to implement supervisors with the existing control system.
- How to verify the decision logic for completeness with respect to the design specifications.
- How to test the implemented supervisor system.

Systematic design of supervisors and the decision logic for fault tolerant control is still an unexplored area, which is also recognized by Patton (1997). A general methodology in a neighbouring field has, nevertheless, been treated by Yazdi (1997), that develops a supervisory control system for the start-up and close-down of a distillation plant. The following introduction, mainly compiled from Yazdi's thesis and the references herein, is not meant as a basis for the methods used within this thesis, but more as a discussion on related approaches.

The expression *supervisor* has been used in different areas with various interpretations:

- *Monitoring*. Man-machine interface for visualization of process state of operation and control performance.
- *High level control and planning*. Management of production plans (start-up, close-down, and set-point generation). Path planning for robots.
- *Controller tuning*. On-line monitoring and tuning of controllers.
- *Fault handling*. Detection and diagnosis of faults with subsequent on-line decision making and reconfiguration.

Yazdi gives a general definition that also applies to the approach taken in this thesis, although it is developed to a slightly different purpose:

Definition 2.3.1 *From Yazdi (1997): A supervisory system is a system that*

1. *evaluates whether an object behaviour satisfies a set of specified performance criteria,*
2. *diagnosis causes for lack of performance fulfillment,*
3. *plans actions, and*
4. *executes planned actions.*

Item three involves the task of decision taking, which is the subject of this subsection.

A formal approach to the design of the decision logic is developed in connection with control and supervision of *discrete event systems* (DES). The term "supervisor" is used by the DES community to cover the same functionality as the decision logic in FTC. Modelling of the behaviour of DES systems is event-driven as opposed to the conventional differential/difference equations that are time-driven. Typical DES applications are telephone networks, flexible manufacturing systems, and batch processes. Although fault events basically exhibit a DES behaviour, this thesis does not consider formal DES theory for the synthesis of supervisors. Recent developments in DES theory relevant for FTC include Overkamp (1997) that describes a method for handling nondeterminism and Cho and Lim (1997) that treats failure analysis and diagnosis in dynamical systems.

A formalism called *supervisory control theory* (SCT) has been developed within the DES community (Wonham (1988)), where the process is modelled as a *finite state machine* (FSM) with controllable and uncontrollable transitions. The supervisor is realized as a similar FSM, designed with a subset of the model-FSM describing the desired (or legal) states. The supervisor then disables the controllable transitions in such a way that the process is forced into the desired states. Within this paradigm SCT provides a formal specification language, formal proofs for controllability, and existence of supervisors (Cassandras *et al.* (1995)).

A shortage of SCT is that the supervisor only facilitates disabling of transitions and provides no mechanisms for actively enforcing transitions. A theoretical modification of SCT that overcomes this problem has lead to *procedural control theory* (PCT) defined by Rotstein *et al.* (1995). Within this framework procedural controllers can be synthesized to make sequential control of DES processes for normal operation as well as abnormal operation.

Along another track the graphical tool *Grafcet* has been developed, based on Petri Net theory, as a platform for implementing sequential as well as parallel logic controllers (David and Alla (1992)). Grafcet is very powerful for modelling controllers, but - in contrast to PCT - informal and provides no formal proofs.

A combined usage of PCT and Grafcet is suggested in Yazdi (1997) to make a *completeness check* of the supervisor with respect to handling of abnormal situations. A

mapping from Grafcet into the FSM domain, used by PCT, combines the power of each method: Flexible controller specification in Grafcet and formal proofs in PCT. A logic controller specified in Grafcet can then be translated into FSM and verified against an FSM-model of the system for completeness.

This thesis does not make direct use of SCT, PCT, or Grafcet, although some similarities exist. It is a key subject in SCT and PCT to provide a formal approach to synthesis of the decision logic of supervisors, but formal methods are not in focus in this thesis. In this thesis the decision logic is implemented with *state event machines* (SEMs) which is equivalent to FSMs. Completeness check is also considered in this thesis, but instead of modelling the process as an FSM with faults causing uncontrollable transitions, faults are modelled in a logic network reflecting their *propagation* through subsystems. In this framework it is possible to verify completeness of the reconfiguration capabilities of the decision logic with respect to possible failure modes.

The basic philosophy in this thesis is to provide guidelines for FTCS design which do not require the design engineer to get acquainted with advanced mathematical disciplines that exceeds the normal capabilities of control engineers in industry. This makes SCT and PCT unappealing, also because the FSM description exhibit an exponential growth in complexity with increasing plant complexity. Furthermore, the formal specification for SCT has proven to be very difficult to establish from natural requirements, although dedicated design techniques exist (Cassandras *et al.* (1995)). On the other hand, formal methods for existence and completeness are attractive, so the usage of Grafcet (or similar tools) in connection with PCT or SCT is an interesting area for future research.

2.4 Summary

Design of FTCSs is a discipline within reliable systems development where the presence of faults is detected on-line and accommodated automatically. In contrast to fail-safe systems for high risk applications, an FTCS for ordinary industrial processes accepts graceful degradation of performance, so it is not required to include a high level of sensor and actuator redundancy. Instead, the control system is designed to increase availability as far as possible by using available instrumentation in a clever way. This thesis presents a development methodology for the design of reliable FTCS software that fits into existing software development strategies, and practical guidelines are provided, based on experience from case studies. Existing techniques for fault detection, isolation, and reconfiguration are combined with a systematic analysis and design strategy that has the purpose to guarantee a consistent implementation of the supervisory system. It is assumed that the FTCS software is installed on a reliable platform, so the entire system can operate satisfactorily.

Chapter 3

Development Method for Fault Tolerant Control

This chapter describes the ingredients of fault tolerant control systems and how the building blocks are combined into an implementational architecture, used throughout the thesis. The architecture has been developed by the department (Blanke *et al.* (1997)) with the purpose to facilitate simple design, reliable implementation, and systematic testing. A general development procedure is presented that matches this architecture. This development procedure was initially considered in Blanke *et al.* (1993) and has been matured in connection with the satellite FTCS design (Bøgh *et al.* (1995, 1997)). The purpose is to give the design engineer a set of coherent activities that, when combined, guarantees completeness of fault coverage with respect to the design requirements. This chapter illustrates the general development procedure by applying it to the design of an FTCS for the benchmark equipment. This case study was earlier introduced in Blanke *et al.* (1993). The benchmark design is based on the development method and highlights important aspects of real industrial applications.

3.1 General Supervisor Architecture

The concept of fault tolerance in control systems basically means that the traditional feedback controller is augmented with an algorithm that detects and locates faults, determines a proper remedial action, and reconfigures the system to accommodate the fault. The *system* in this context is considered as a local control loop or a group of connected subsystems, where information from different sources can be processed to make sensible decisions. *Faults* are considered as anomalous behaviour of components that can possibly lead to a complete failure of a subsystem or the entire system. A *reconfiguration* of the system should prevent the fault from developing into a failure.

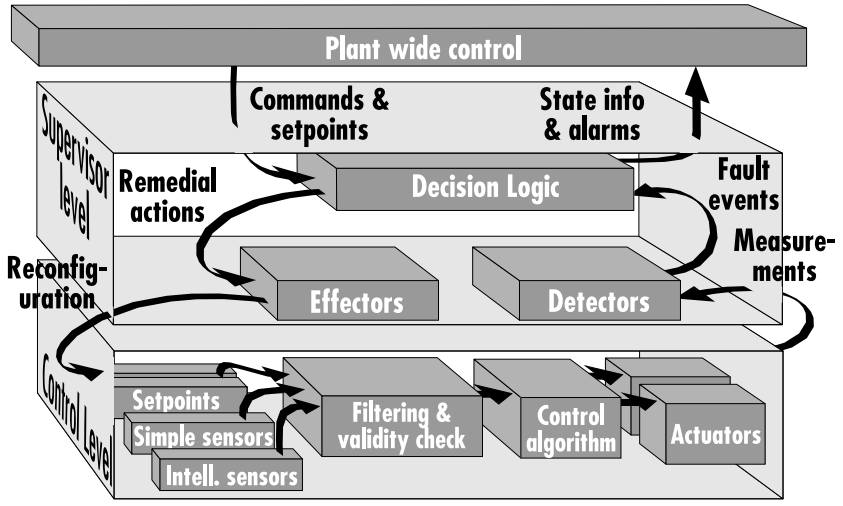


Figure 3.1: *A general three level architecture for fault tolerant control systems. The three levels are feedback control, fault detection and reconfiguration, and supervision decision logic. The latter also has communication to the plant-wide control system.*

The fault handling functionality is combined with the traditional feedback control system and also communication with higher levels as shown in Figure 3.1. The fault *detectors* are signal processing units that use any available process signal (measurements and internal variables) to detect anomalies in the control level. A successful detection and isolation of a fault is reported to the *decision logic* as a boolean *fault event*. The decision logic determines an appropriate *remedial action* based on the current operational state, which is known by the supervisor. The supervisor also includes the command and monitoring interface connected through a *plant wide control* system. The actions requested by the supervisor are executed by *effectors* that perform the actual *reconfiguration* in the software of the system.

The assignments of the three levels are described in the following sections.

3.1.1 Control Level

The control level consists of the conventional feedback control loop with sensors and actuators. Single sensor sanity check (limit, rate, mean value checks, etc.) and basic filtering (smoothing, EMC protection, outlier removal, etc.) are included as inherent elements of any FTCS. The validity check has the purpose to prevent propagation of invalid sensor data into the controller and further to the actuators. This sensor status information can also be used by the fault detectors and the supervisor for further analysis.

3.1.2 Detector-Effector Level

The purpose of the *detectors* is to detect single faults in all components including sensors, actuators, and networks. They are implemented as a number of units that each detects and isolates one fault or a coherent group of faults. The fault detectors are able to detect more faults than the sanity check in the control level, because they employ knowledge about the relation between several signals. The detection of faults can be achieved in different ways, such as voting between hardware redundant sensors (e.g. Tzafestas and Watanabe (1990)), statistical tests on signals (e.g. Basseville and Nikiforov (1993)), and comparison between known process behaviour and measured behaviour (see the survey in section 2.3.2). The latter, which is in focus in this thesis, utilizes analytical redundancy in the process where fault events can be identified by means of the general procedure shown in figure 3.2. Input and output measurements are processed using a

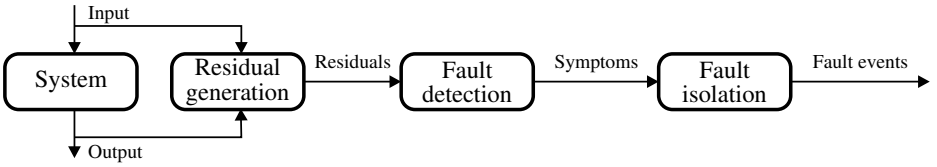


Figure 3.2: *The process of determining fault events requires the three steps of residual generation, fault detection, and fault isolation.*

model of the system to generate *residuals* that highlight discrepancies. Changes in the residuals are then detected and used to generate *symptoms* on the event. Further diagnostic facilitates isolation of the actual failed component and produces *fault events*. The signal processing performed in these steps is also able to produce characteristics like fault type and time of occurrence. This diagnostic information is also made available for the decision logic. A brief overview of methods for fault detection and isolation (FDI) was given in section 2.3.2. A detailed study on several algorithms for model based FDI is given in chapter 4 in connection with the benchmark application.

The *Effectors* are the second block at this level. They include an appropriate reconfiguration algorithm that changes control of the plant to prevent the detected fault from developing into a failure. The reconfiguration can be simple switching between hardware components, but also more complicated methods like on-line controller redesign. An overview of approaches in this field were presented in section 2.3.3.

3.1.3 Decision Logic Level

The decision logic has two objectives; first it shall determine the best modification of the system to accommodate a given fault and secondly, it shall provide information about alarms and the operational state to the plant wide control system. The decisions are taken on the basis of the following information:

- *Operational state*. The decision logic keeps track on the operational state of the control level (sensor and actuator configuration, controller mode, etc.).
- *Fault events and fault information*. The decision logic receives fault events that either represent true alarms or false alarms about faults. Additional information as statistics on the number of alarms can be used to diagnose faults. Supplementary fault information from the detectors as for example time of occurrence and fault type can also be used in the deduction of an appropriate remedial action.
- *Command inputs*. The decision logic receives high level commands from the plant wide control system. Commands can be directed to the control level (start-up, close-down, and set-point changes) or to the configuration of the supervisor system (e.g. enabling and disabling of a fault detector).

The decision logic is adequately realized as a set of *state-event machines* (SEMs) implemented as shown in figure 3.3. The SEMs map inputs (fault events and com-

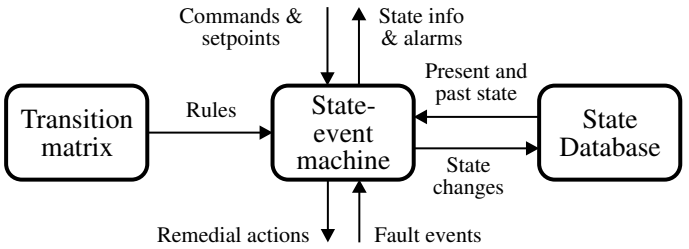


Figure 3.3: *The supervisor decision logic realized as state-event machines. The decision logic uses a state transition matrix to deduce an appropriate remedial action based on fault events, command inputs from higher levels, and knowledge about the present and past state of operation.*

mands) to outputs (remedial actions and state/alarm information) depending on present and past states given in a *state database* and a set of rules given in a *transition matrix*. The structural separation of the rule base is advantageous because it can be verified for correctness without considering the implementational problems of the SEM inference.

3.2 Fault Tolerant Control System Development Process

This section presents a guide on how to design the three blocks in FTCS, introduced in figure 3.1 in the previous section: *detectors*, *effectors*, and *decision logic*. The instrumentation and controllers of the control level are assumed to be already designed, so the approach takes its starting point in the analysis of an existing system.

The development strategy is a systematic approach that has the objectives to

- facilitate complete coverage of possible single faults,
- support means for consistent and complete specifications for the plant reconfiguration algorithms based on a fault propagation analysis,
- make automatic completeness and correctness tests on the final supervisor decision logic design,
- avoid unnecessary plant modelling,
- use software tools for analysis, and
- use automatic code generation.

These objectives are in focus in the strategy outlined in figure 3.4. These eight steps

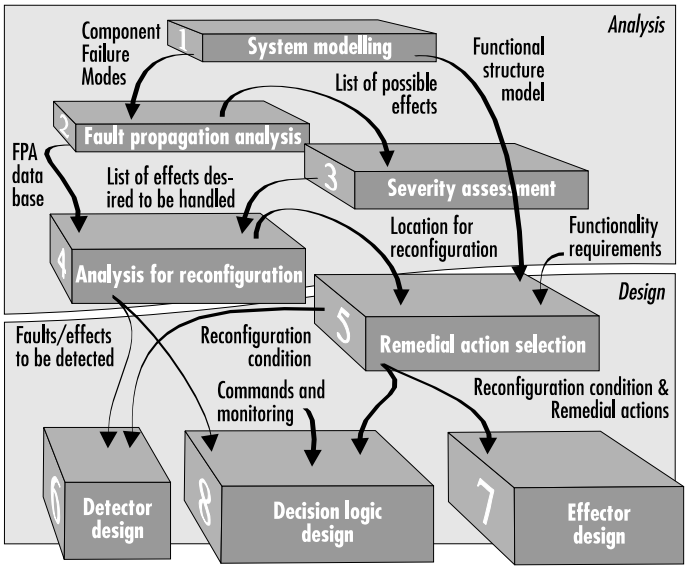


Figure 3.4: *Systematic fault tolerant control system development approach. These eight steps conducts the designer from an analysis of failures to the design of fault detectors, supervisor decision logic, and fault accommodation.*

should be applied in the development of any control system, where the advantages of increased performance and availability from FTC can be exploited.

The analysis begins with a *system modelling*, where individual components are identified and subsystems are organized into a functional structure. Potential failure modes

of each component are identified. An analysis of the *fault propagation* (FPA) from each component and throughout the entire system in the functional structure is then performed. This provides a list of fault effects that are judged for *severity* according to their influence on system performance. The end-effects that are found severe enough to demand attention are identified. The FPA database from the first step is then used to create a fault tree, where faults-effects relationship can be analyzed. The exact location in the process where the propagation of each fault can be stopped by *reconfiguration* is identified from this fault tree. This information is then used together with the functional structure model to select a set of *remedial actions* that have the ability to reconfigure the process according to the requirements for fault tolerance. The purpose of the analysis so far is to provide information for the definition of fault detection, isolation, and reconfiguration (FDIR), i.e. faults/effects for the *detector design* and remedial actions for the *effector design*. The detector and effector design tasks involve complex numeric analysis and represent the major workload. The benefit of the initial qualitative analysis is that it provides exact requirements for FDIR, i.e. identifies which faults shall be detected, which faults shall be isolated, and where the plant should be reconfigured. This helps to avoid excessive FDIR design. The last step is the design of the *decision logic* that shall select a suitable remedial action when a fault has been detected and also takes care of command and monitoring.

The eight steps fit into the general water-fall model described in section 2.3.1 as shown in figure 3.5. The first considerations about faults are included already during

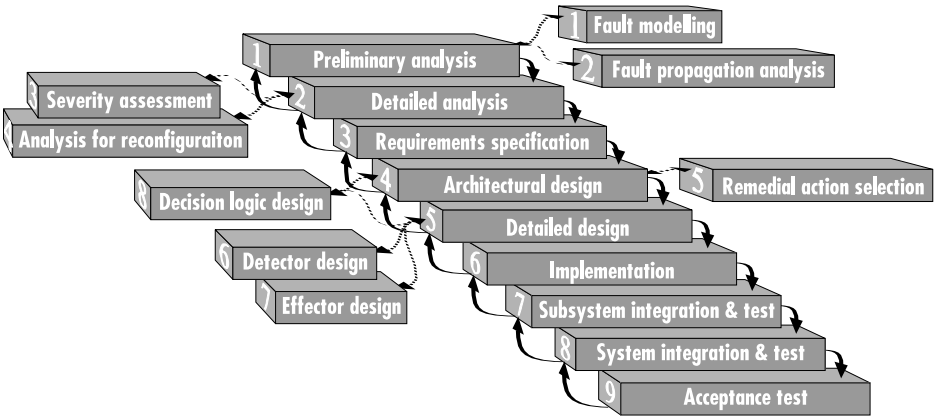


Figure 3.5: *The specific actions for FTCS design, shown in figure 3.4, shall be included in the general water-fall development procedure as shown in this figure.*

the preliminary analysis. When more system knowledge has been collected during the detailed analysis, then the fault effects severity can be estimated and possibilities for stopping the fault propagation identified. The set of remedial actions shall be determined

during the architectural design, where also the overall design of the decision logic can be performed. The detailed design of the decision logic can be completed when the detectors and effectors have been designed.

A further explanation of the individual steps in this development method is the subject of the remaining part of this section. The next section demonstrates the procedure with a complete design of an FTCS for the benchmark application.

3.2.1 System Modelling.

The process of FTCS development begins with a systematic high level modelling of the system under concern. The model needs only be descriptive for the behaviour under faulty conditions, which allows it to be simple compared to dynamic models used in, for example, simulation and control. It is the purpose of the two initial steps (fault modelling and fault propagation) to establish a *qualitative* model of how the system reacts to faults, that can be used for the subsequent analysis and design steps.

Hierarchical system model. The fundamental requirement to the modelling paradigm is that it must be intuitively simple and it must capture all kind of systems (continuous/discrete time, mechanical/electrical, continuous/logical variables, etc). The choice is basically between *functional* models and *fault* models. The functional models describe the *correct* behaviour of the system whereas the fault models describe the behaviour in *fault* situations. Functional models are typically based on analytical relationship between process variables and are often very complex. Hence, they should be avoided as far as possible because they require superfluous effort. Fault models include only the necessary information required for describing the effect of a fault. They describe failure modes and fault propagation in logic terms, which is advantageous when modelling different kind of systems. The disadvantage with fault models is that each failure mode must be specified. In functional models, all other modes than normal are automatically failure modes. This means that fault models have two drawbacks; first, they do not capture unanticipated faults, and secondly, the analysis becomes complicated when the components have a large number of failure modes. Fault models are used as the basic approach in the following, but functional modelling may be necessary if the relationship between a fault and the effect on some subsystem is too difficult to determine in logic terms.

An additional requirement is that the description of the system structure must be adequate for modelling the propagation of faults through subsystems to system level. The structure of a complex system can be captured in a hierarchical model in various ways:

1. Component hierarchy, where the physical components are grouped by type (mechanical/electrical, digital/analog, etc).

2. Physical structure model, where the physical components are grouped by location

and described by their interconnections.

3. Functional structure model, where the system is broken down into units that perform sub-tasks.

The *component hierarchy* is mentioned here only as a counterpart to the others because it is suitable for implementation using object oriented techniques. A case study on the usage of a component hierarchy for an automated failure mode and effects analysis in hydraulic systems is presented in Atkinson *et al.* (1993). The paper presents an expert system that combines component objects at run-time, thus facilitating re-usability of objects.

The *physical structure model* describes component failures in terms of *failure modes* whereas the *functional structure model* describes functionality failures in terms of *discrepancies*. A discrepancy is characterized by an abnormal behaviour of a physical value (out-of-range, oscillation, etc.) or violation of a relation between physical values (e.g. if a motor wire disconnects, the discrepancy could be: "Shaft velocity zero although motor voltage non-zero"). A discrepancy can also describe a degradation of the safety system, in which case there may be no immediate effect on the system operation, but continued operation has become unsafe. The two modelling techniques are used in combination in this thesis to give a suitable modelling paradigm. The required level of detail in the hierarchy is determined, for each case, by the design engineer based on the complexity of the subsystems and the requirements and possibilities for reconfiguration. In principle, the analysis can continue down to the level of capacitors and resistors. But if some subsystem has entirely known failure modes, known reaction to faults on inputs, and the entire subsystem will be substituted on failure, it may not be necessary to perform a detailed breakdown. The use of two different principles may seem superfluous, but it will show up later in the thesis to be advantageous. The physical model is used to locate possible component faults and the functional model is used to determine the propagation of faults. The association between the two models can be used to verify if a particular reconfiguration scheme actually stops the propagation of faults. It will be clear from later examples how the concepts are utilized.

Fault modelling. The physical model is used to identify possible component faults. Traditionally, roughly three techniques have been used: preliminary hazard analysis (PHA), hazard and operability studies (HazOp), and failure mode and effect analysis (FMEA). PHA is an early identification method providing a coarse inspection of faults. HazOp covers both fault analysis and operational problems, but suffers precision and is thus not suited for completeness of fault coverage. FMEA is very detailed, but is poor in identifying interconnected failures between subsystems. A further discussion of the three techniques can be found in Bell (1989). This thesis applies the FMEA method, which is well documented in form of standards and manuals (DoD (1980); Ford/General Motors/Chrysler (1995)), extension to software FMEA (Lutz and Woodhouse (1996)), and automation for electrical circuit design (Price *et al.* (1997)). A graph-

ical representation, called *matrix FMEA*, was used as basis for earlier publications in the department: Blanke *et al.* (1993), Blanke and Jørgensen (1995), Bøgh *et al.* (1995), Jørgensen (1995), and Blanke (1996). It was introduced by Barbour (1977) and later automated in software by Legg (1978) and Herrin (1981). The matrix method is actually used to describe the propagation of faults, and this has inspired the approach taken in the thesis. The modelling of fault propagation is covered in the next section.

The FMEA technique is used, because it is good to ensure that all possible faults are considered. It is based on a standardised documentation formula that pushes the design engineer to make a complete analysis where each fault shall be described by failure mode (type of fault), failure cause, failure effect, frequency, and to which component/subsystem the fault is associated. The FMEA is combined with the fault propagation analysis, that is powerful in describing the interconnections between subsystems.

The number of faults to include in the analysis is decided by the design engineer from detailed system knowledge, price for additional complexity, probability of failure, and information from reliability databases (if available).

Once the failure effects have been determined by the FMEA, the physical structure model is linked together with the functional structure model and propagation to functional discrepancies are determined. This is the topic of the next section.

3.2.2 Fault Propagation Analysis

The purpose of the FPA is to examine how the component failure modes, worked out from the FMEA, propagate through the *functional structure model* to end-effects on system level. The result of the FPA is a fault propagation graph (FPG) that shows the dissemination of discrepancies through each subsystem in the functional structure hierarchy. The FPA-method presented in this section is based on earlier work (Blanke *et al.* (1993); Blanke and Jørgensen (1995); Bøgh *et al.* (1995); Jørgensen (1995); Blanke (1996)), but further developed to be more flexible and applicable for a wider range of systems.

Fault propagation graphs. The list of component failure modes from the previous section contains a finite number of *effects* associated to physical values. The functional model describes how these effects turn into discrepancies and how the system reacts under these conditions. In feedback control systems, which are in focus in this thesis, faults often cause values to go to extremes or become constant. This can, for example, be modelled as a three-tuple (Low, High, Constant). As an example, the logic model for a potentiometer could be the multi-value truth table shown in table 3.1. The first three rows represent non-faulty operation, where the position measurement equals the actual position. The last three rows represent the three failure modes where either the negative wire, the positive wire, or the output wire is disconnected. All these faults cause the output to be independent on the input, which is written as '-' in the "Input Position" column.

Table 3.1: *Logic model for a position potentiometer.*

Potentiometer	Input	Output
Fault	Position	Position measurement
NoFault	Low	Low
	Constant	Constant
	High	High
VnegWireDisc	-	High
VposWireDisc	-	Low
VoutWireDisc	-	Constant

All components are modelled in this way and connected into a multi-valued logic graph, where the operations can be a combination of any logical operations (AND, OR, and NOT). In this thesis the multi-valued FPG is analyzed using the Beologic™ array inference tool-box (AIT) from Bang & Olufsen, Denmark. This program is only able to apply boolean logic, but it is always possible to translate multi-valued logic into boolean logic with a penalty on memory consumption. The software is dedicated to analyse logical circuits, where logic relations are written as rules (Møller (1995)). It compiles the rule base into a special representation using *array logic* theory, introduced by Franksen (1979). This makes the run-time inference very fast and assures a fixed response time. Beologic™ is able to analyse the logic relations between the involved variables, which is very helpful for the purpose of this thesis. As an example on the use of boolean logic, the above potentiometer truth table can be translated into the Beologic™ rules shown in table 3.2. The description of the faults propagation in Beologic™ is based on

Table 3.2: *Beologic™ rules for the potentiometer truth table in table 3.1.*

#	Rule
0	`NoFault = not (VnegWireDisc or VposWireDisc or VoutWireDisc)`
1	`PosMeasLow = ((NoFault and PosLow) or (not NoFault and VposWireDisc))`
2	`PosMeasConst = ((NoFault and PosConst) or (not NoFault and VoutWireDisc))`
3	`PosMeasHigh = ((NoFault and PosHigh) or (not NoFault and VnegWireDisc))`
4	`OneOf (PosMeasLow PosMeasHigh PosMeasConst)`

the *bi-implication* (=) that constrains both the left side and the right side of the rule. The alternative, used in Jensen *et al.* (1994), is to use an *implication* (<-) which only constrains the consequent (left side). If only the implication is used, problems arise when the rule base is examined for causality backwards from end-effects to faults, because the antecedents (right side variables) are not constrained by the rule. The last rule ensures that one and only one of the three values of the output variable will be true.

Feedback loops in graphs. A special problem in FPA is the treatment of *feedback loops* where the logical feedback may cause contradictions in the FPA rule base. As an example, consider the feedback loop illustrated in figure 3.6. The fault variable f causes

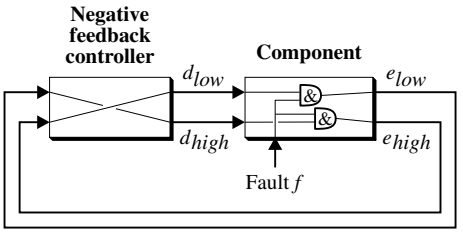


Figure 3.6: *An example on fault propagation in a system with negative feedback.*

the effect variable e to be either e_{low} or e_{high} depending on the value of the discrepancy d. The value of d is determined by e such that there is a "cross-over" which is caused by the sign change in the negative feedback controller. The FPG is represented by the following rules:

$$e_{low} = f \wedge d_{low}, \tag{3.1}$$
$$e_{high} = f \wedge d_{high}, \tag{3.2}$$
$$d_{low} = e_{high}, \tag{3.3}$$
$$d_{high} = e_{low}. \tag{3.4}$$

The values of the logic variables are furthermore mutually exclusive which cause the following constraints:

$$d_{low} = \overline{d_{high}}, \tag{3.5}$$
$$e_{low} = \overline{e_{high}}. \tag{3.6}$$

A boolean reduction of, for example, e_{low} gives

$$e_{low} = f \wedge \overline{e_{low}}. \tag{3.7}$$

This equation is only valid if both f and e_{low} are false. This means that the FPG is not representative for the behaviour of the physical system and it is not possible to analyse the end-effects from the corresponding fault using the FPG. It is therefore necessary to first identify such fault inputs and then treat them separately.

The examination of large and complex FPGs for the identification of fault inputs *bound to false* is adequately supported by Beologic™. For each combination of external conditions (set-point etc.), Beologic™ can perform the boolean reduction of the FPG rule-base and generate a list of variables bound to false. The FPG is then modified by decoupling the identified fault inputs from the loop and instead assign them to a dedicated "oscillation" discrepancy. All involved logic models are augmented with an oscillation discrepancy to describe the propagation of the oscillation.

In some cases, a logical oscillation in the FPA analysis is associated with an unstable behaviour of the underlying physical system. One example is a disconnected tachometer wire within the position control loop of the benchmark equipment described in section 3.3.3, where the fault causes the output arm to oscillate around the desired position reference.

The issue of feedback loops is also treated in Blanke (1996), Jensen *et al.* (1994), and Nilsen and Blanke (1996). The approach taken in Blanke (1996) is to consider the cases where feedback loops do not lead to potential oscillations in the physical system. This means that there are no problems with analysis of the corresponding FPG. The solution presented in Jensen *et al.* (1994) is to remove the feedback and consider the different paths of the FPG separately. This means that, in the example in figure 3.6, the fault f will have two effects, e_{low} or e_{high}, depending on d. The discrepancy d is then considered as an additional input to the analysis. This means that potential oscillations in the physical system are not specifically analysed. Nilsen and Blanke (1996) proposes to open the loop at a suitable point and then leave the remaining analysis to the design engineer.

Modelling of controllers. Another issue is how to design a logic model of a feedback control law. It is only necessary to describe two scenarios; first, the normal behaviour where there is no fault in the loop and second, the behaviour where a fault causes the loop to malfunction. As an example, the truth table of the logic model for a negative feedback position controller is shown in table 3.3. The first three rows describe the

Table 3.3: *Logic model for a negative feedback position controller.*

Feedback controller	Inputs		Output
Fault in loop	Position measurement	Position reference	Desired velocity
No	-	Low	Low
	-	Constant	Zero
	-	High	High
Yes	Low	-	High
	High	-	Low
	Constant	Low	Low
	Constant	High	High

situation with no faults in the feedback loop, where the output equals the reference value and the feedback is neglected. The fourth and fifth row depicts the cases where some loop fault causes the measured value to go either low or high. The output is now independent of the reference, because the measured value is either lower or higher than the reference value. The last two rows represent the case where a loop fault causes the position measurement to get stuck. In this case, the output value depends on whether the reference is lower or higher than this constant measurement value.

A technique has now been presented that provides means for automated analysis of a

network of interconnected logic subsystem models. The output is a list of end-effects on system level and an FPG that describes the relation between fault inputs and end-effects. The FPG has no problems with conflicts arising from logic feedback loops, because these cases have been organized into separate propagation paths that are independent from the feedback loops.

3.2.3 Severity Assessment

The next step in the development process is to judge the severity of the end-effects, i.e. to categorize the impact on the systems operation from the consequences of the faults. The purpose is to select which end-effects *should* be treated and which *can* be treated with existing system components.

In reliability engineering the evaluation of severity can be included in the failure mode and effects analysis as a *criticality* assessment on the component failure modes. Criticality is a combined measure between severity and probability of occurrence (see DoD (1980) for a mathematical definition). It is often difficult to quantify criticality on a component level, because a fault's consequence on system level is not known. An alternative approach is chosen in this thesis (see also Blanke *et al.* (1993) and Bøgh *et al.* (1995)), where the severity assessment is performed directly on the end-effects associated with component faults. Probability of occurrence is not considered at this point, because all faults included in the analysis are regarded as equally important. Failure modes, that are clearly irrelevant, were omitted in the FMEA in section 3.2.1.

The end-effects are categorized into *hazard classes* that defines how serious the consequences are to aspects like diversion from desired operation (e.g. product quality), damage to equipment, damage to environment and people, increased wear on equipment, increased cost of production, degradation of the safety system, etc. It is often necessary to apply some engineering knowledge about the system to estimate the severity of an end-effect, i.e. how big the effect is and how fast it develops. The severity assessment must, therefore, be performed in the detailed analysis phase.

A formal classification of the end-effect severity has the benefit that if the system is modified or the requirements are changed, the designer can easily determine a new set of end-effects that requires accommodation.

3.2.4 Analysis for Reconfiguration

At this point in the development process a list of end-effects, that must be handled, has been created. The next step is to locate the components that can be reconfigured to stop the propagation of faults which cause these end-effects.

This is achieved by organizing the logic models into a *fault tree*, which is a network of OR and AND operators that describes the causal relation between faults and discrepancies. Figure 3.7 shows the fault tree of the logic models from table 3.1 and table 3.3. The analysis of the fault tree is performed with two purposes; first, it shows where the fault propagation can be stopped by a certain component reconfiguration and second, it

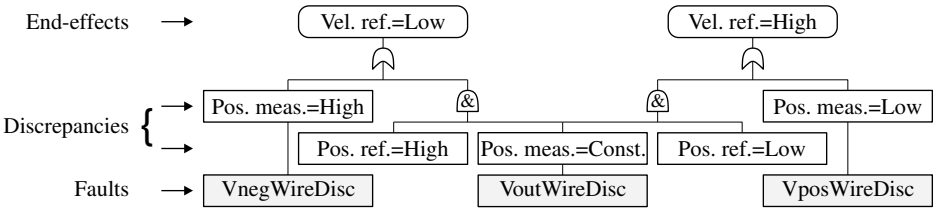


Figure 3.7: *A fault tree of the fault propagation for the position measurement and control from table 3.1 and table 3.3.*

supports the fault detector design with information about which measurement(s) can be used to detect and isolate the event (fault or discrepancy) that is needed for the reconfiguration. The fault tree can be automatically generated from the FPG rule base, which is advantageous when a manual analysis becomes unmanageable. Feedback loops are not a problem in the graph, because the faults that cause unwanted constraints in the graph have been arranged into independent branches in the tree as described in section 3.2.2.

3.2.5 Remedial Actions Selection

The next step in the development of an FTCS is the identification and selection of remedial actions that are able to reconfigure the plant to oppress the propagation of potential faults. This process is assisted with the fault tree produced in the previous section, but also requires insight into the possibilities for reconfiguration of the process. It is not the purpose, at this point in the development process, to design the actual fault accommodation algorithms. This is postponed to after the FDI design, so it is certain that an FDI algorithm exists which provides the information required to trigger the remedial action.

The task of locating possibilities for reconfiguration of the system is, by nature, extremely application dependent. It is difficult to give general advises, but it is advantageous to use the fault tree worked out in the previous step. The fault tree can be used to point out which subsystems to examine and where to look for reconfiguration possibilities. It also shows, if a discrepancy can be triggered by a group of faults. In this situation it may be possible to detect the particular discrepancy and reconfigure the system to accommodate all faults in that group in one single remedial action. It is up to the design engineer, whether to use the fault tree as a starting point, or to consider the abilities of the system first and then use the fault tree to determine if the fault propagation will be stopped successfully. In either approach, the fault tree should be examined to verify if all faults that shall be handled actually *are* accommodated. It may not be possible to handle all faults, in which case, either the plant must be augmented with more capabilities or the requirements must be relaxed.

In the search for remedial actions, the following possibilities can be considered:

- Reconfiguration between redundant hardware. If redundant sensors and actuators

are available, continued operation with full performance is possible.

- Entering a degraded mode. It may be possible to disable the unhealthy subsystem and continue operation in a degraded mode.
- Off-line reconfiguration of controllers. Change the configuration of sensors/actuators and controllers to avoid using failed components. A set of controllers is designed off-line to meet anticipated failures.
- On-line redesign of controllers. This is state-of-the-art in reconfiguration where an on-line algorithm is executed to modify control parameters based on the present fault scenario. This facilitates possible coverage of unanticipated faults, but is also subject to a very complex (and therefore risky) design.
- In case of a reference set-point fault it may be sensible to restore the situation to the last non-faulty value and then continue control.
- Close-down is the last resort if no other possibilities exist.

A short survey of methods for on-line and off-line reconfiguration were given in section 2.3.3.

Once a remedial action has been selected, it is possible to specify the corresponding *reconfiguration condition*. The reconfiguration condition defines how serious the effect of a fault is allowed to be before the reconfiguration must be performed. This has often been formulated as a temporal requirement (especially by the fault detection community), but it is much more sensible to define it with respect to the impact on the systems operation. Such a criteria is typically a variable or discrepancy that exceeds a threshold, a maximum time spent in an undesired operational state, or some statistical property of a specific behaviour.

The reconfiguration conditions are used to derive requirements for the fault detection and accommodation design. The reconfiguration condition is used to derive the requirement for the sensitivity of the fault detector and it also indicates how fast the reconfiguration must be performed in a worst case situation. The specification of the reconfiguration condition is based on the end-effect severity assessment (section 3.2.3) and insight into the possibilities of the process.

3.2.6 Fault Detection and Isolation Design

This section describes how the design of the FDI algorithms included in the *Detectors* module in figure 3.1 relates to the development steps performed so far. It is illustrated how the requirements for the design are derived from the above analysis.

The first action is to determine a proper FDI method that matches the particular problem. It is up to the design engineer to select a suitable method among the techniques described in section 3.1.2 and from the survey in section 2.3.2.

Once an FDI method has been selected, the FPG is used to examine the necessary requirements for isolation between faults. It is very important to check if a specific detector is sensitive to other faults than the expected. It is unacceptable to perform a reconfiguration that does not match the actual fault. The issue can be investigated by expanding the FPG with the functionality of the fault detectors. The rule base can then be searched for possible faults that trigger the detectors. This list is then inspected to check if it is necessary to distinguish between more faults and include additional detectors. The major difficulty of doing this analysis in the FPG framework is to ensure agreement between the fault detectors and the physical characterisation of the discrepancies they are supposed to detect. The problem is two-fold: First, when several failure modes are modelled to cause the same discrepancy, care should be taken to ensure that each of them is correctly described by this discrepancy. Secondly, the FPG is not originally designed to describe *detectable* discrepancies so a suitable discrepancy may not exists for a fault detector. If this is the case, it may be necessary to augment the graph with more discrepancies. When these problems have been solved, the usage of the FPG gives a clear overview of which faults must be treated individually and which can be considered together as a group.

The next action is the actual design of the fault detectors. Depending on the above choice of method, it may be necessary to derive mathematical models of specific subsystems and obtain deeper insight into the system and the characteristics of the failure modes. The problematic issue of fault detector design is always the trade-off between false alarms and correct detections. The primary requirement for a fault detector is the sensitivity to faults, which is derived from the *reconfiguration conditions* given in the previous section. False alarms come from unknown inputs like disturbances and noise and also from modelling errors (e.g. nonlinearities). FDI algorithms must be designed to be *robust* against these exogenous signals and this topic has attracted a large interest in the FDI community. The design of analytical redundancy algorithms is covered in more detail in chapter 4, where different issues will be discussed in connection with the benchmark problem.

In the design of fault detectors, it is important to be aware of potential problems with initialization of the filtering algorithms, which may occur in connection with start-up of the system or after a reconfiguration has taken place. It can be advantageous to let the supervisor's decision logic handle the overall control of these problems, so it is important that they are recognized at this stage.

3.2.7 Fault Accommodation Design

This section briefly outlines how the design of the fault accommodation algorithms, included in the *effector* box in figure 3.1, fit into the development methodology. The fault accommodation algorithms execute the procedures associated with the remedial actions that are requested by the supervisor decision logic.

The remedial actions were selected in step 5 in section 3.2.5. At the present step, the

advanced algorithmic design of the methods for controller reconfiguration and on-line redesign takes place. These methods require deep insight into the characteristics of the system, so controllability and stability of the new controllers can be assured. It is out of the scope of this thesis to consider these aspects, so the reader is referred to the survey in section 2.3.3 and the references herein.

One requirement, derived from the previous analysis, is the *reconfiguration requirement* defined in section 3.2.5. This requirement determines how long time the fault accommodation algorithm is allowed to run before the control system is reconfigured.

Other aspects of fault accommodation design are the problems of start-up/close-down of the system and bump-less transfer between different control schemes when the system is reconfigured. These issues must be considered together with similar problems of the fault detectors, designed in the previous section, so a suitable enabling/disabling scheme of the detectors and effectors can be set up for the supervisor decision logic.

It is, furthermore, a risk with on-line redesign algorithms, that the objectives of the redesign cannot be met in a particular situation. These algorithms, therefore, include a measure of success that is fed back to the supervisor decision logic, so a suitable backup action can be executed if the primary remedial action failed.

These problems of fault accommodation design are not present in the application examples of this thesis, so they will not be treated further on.

3.2.8 Supervisor Decision Logic Design

The last step in the development of fault tolerant control systems is the design and implementation of the supervisor *decision logic* module from figure 3.1. This section illustrates how requirements to the design are obtained from the above analysis and how the implemented decision rules can be verified for correctness and completeness. Earlier results on the design of supervisor decision logic are available in Bøgh *et al.* (1995), Zamanabadi *et al.* (1996), Bøgh *et al.* (1997), and Blanke *et al.* (1997).

Although this thesis focuses on fault handling, the decision logic shall also manage the information exchange with higher levels in the process (command and monitoring). This is included, at this point, as additional requirements to the design because it is deeply integrated with fault handling. The complete set of requirements to the decision logic can then be summarized as the following:

- *Fault handling*. The requirements for decision making in connection with fault handling are directly given by the remedial action list from section 3.2.5. The *faults* are inputs to the decision logic and the *remedial actions* are required outputs. The list of faults must be checked up against the experiences from the fault detector design in section 3.2.6 as there may be additional requirements for the decision logic. The FDI algorithms may have been designed in such a way, that the decision logic shall perform further diagnosis to isolate between faults and remove false alarms. Also special actions in connection with initialization of the FDI algorithms must be considered. Similarly, there may be additional require-

ments for bump-less transfer during reconfiguration as recognized during the fault accommodation design in the previous section. Finally, the on-line fault accommodation algorithms may include a measure of success of the reconfiguration, which must be handled by the decision logic.

- *Plant wide communication*. The decision logic shall be able to manage commands (set-point changes, operational mode changes, FDIR and monitoring configuration, etc.) from higher levels. It shall also provide monitoring information (current operational state, control and FDIR performance, information about executed remedial actions, command verification messages, etc.) to the plant wide control system.

- *Operational mode control*. There may be different requirements to fault handling and monitoring in different operational modes. The decision logic shall be able to distinguish between these cases.

- *Start-up and close-down of the system*. It is out of the scope of this thesis to consider these topics, but the requirements should be recognized before the decision logic design is finalized. This issue will not be further treated.

The decision logic that fulfils these requirements is now designed and represented by rules similarly to the FPG in section 3.2.2. The purpose is to examine the properties of the decision logic and their interaction with the FPG. In this framework a very powerful test for *completeness* can be carried out. The meaning of completeness in this context is that the decision rules cover all the possible faults that are included in the analysis. The decision rule base corresponds to the transition matrix of the state-event machine implementation introduced in section 3.1.3. An automatic translation of the decision rules into the transition matrix makes this procedure very attractive. Beologic™ is able to support both the analysis of the rules and also automatically generate code for the inference of the rules.

When the decision rules are considered alone (i.e. not combined with the FPG), the following properties can be inspected:

- *Potential reduction in rules*. For complex expressions there may eventually be a more compact way to express the same relationship. It can be advantageous to examine selected subsets of the rule base for these alternatives because a simpler expression is often more clear and therefore less vulnerable to mistakes.

- *Cross examination of rules*. A manual inspection of the relationship between selected variables that appear in different rules can be used to verify correct behaviour.

- *Consistency*. The rules can be examined for contradictions, bounded variables, superfluous variables, and redundancy.

A completeness test can be performed when the decision rules are combined with the FPG together with the logic flow of both fault detection and isolation and also reconfiguration. The logic model of the entire reconfigurable system is then organized as illustrated in figure 3.8. The functionality of the fault detectors and remedial actions are

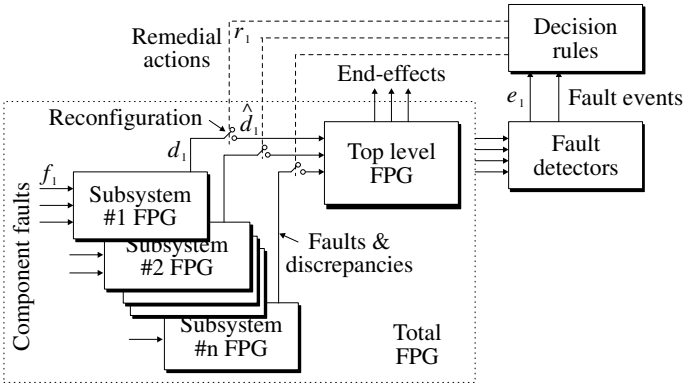


Figure 3.8: *A complete logic model of fault propagation, fault detection, decision, and reconfiguration. This model is used to facilitate a completeness check of the supervisor's decision rules.*

implemented as rules similar to the FPG and decision rules. All necessary information about operational mode dependencies and configuration of the equipment is included so the complete logical behaviour of the reconfigurable system is captured in the model.

The completeness test is based on the properties of this complete logic model. Consider as a simple example the following rules describing a single path in the figure:

$$d_1 = f_1, \tag{3.8}$$

$$\hat{d}_1 = r_1 \wedge d_1, \tag{3.9}$$

$$e_1 = \hat{d}_1, \tag{3.10}$$

$$r_1 = \overline{e_1}, \tag{3.11}$$

where f_1 represents a fault that propagates to a discrepancy d_1. The system can be reconfigured by remedial action r_1 so propagation into $\hat{d}_1$ is prevented. The fault event is detected from $\hat{d}_1$ and sent to the decision logic as event e_1.

A boolean reduction with respect to the propagated discrepancy $\hat{d}_1$ gives the equation:

$$\hat{d}_1 = f_1 \wedge \overline{\hat{d}_1}, \tag{3.12}$$

which is only valid when both $\hat{d}_1$ and f_1 are false. This means that if a fault is successfully detected and the system is reconfigured to stop the fault from propagating, then the

variable representing the propagated discrepancy $\hat{d}_1$ will be *bound to false* by the FPG rule base.

For large and complex systems Beologic™ can support the boolean reduction and be used to examine the following properties of the complete logic model:

- *Fault analysis*. It is possible to analyse that a selected fault is handled successfully by assigning this fault true, setting up conditions (if any), and then inspect the rule base. If there are no solutions, the fault will be handled successfully. Similarly, it is possible to analyse under which conditions the fault will *not* be handled.
- *End-effect analysis*. As for fault analysis, end-effects can be analysed by setting a selected end-effect true and then inspect the rules. If no solutions are found, then the end-effect will never appear. Similarly, it is possible to find the cases, where the end-effect *may* appear.
- *Consistency check*. The fault and end-effect analyses above are performed manually, but it is possible to automate a partial analysis. The faults and end-effects that are designed to be handled in all cases (no conditions) can be verified by a consistency check on the rule base. These faults and end-effects will appear as *bound to false* and can then be verified against the requirements.

An example on this completeness check is given for the satellite application in section 5.6.3.

It should be noted, whatsoever, that the above analyses does not cover system dynamics, so it is not guaranteed that temporal requirements are met. Similarly, the simple logic model prevents analysis of false alarms and missing alarms because signal amplitudes and detection thresholds are not included. The completeness check works on the following conditions: 1) the designer has thought about the faults, 2) it is detected and isolated in time, 3) and the reconfiguration actually stops the fault propagation. Under these conditions it is guaranteed that the decision logic will handle the situation.

3.2.9 Summary

A general development methodology for the design of FTCSs and an implementational architecture have now been introduced. The next sections use the benchmark application to illustrate the individual steps in the development procedure and how the final design can be realized in the three level architecture presented in section 3.1. The benchmark case study focuses on the analysis part, i.e. the FMEA and the FPA and how requirements for the FDI and supervisor designs are derived from this analysis. The fault handling aspect of the benchmark supervisor is rather simple. More complex issues of supervisor decision logic design is presented in connection with the satellite application in chapter 5.

3.3 Fault Tolerant Control System Design for the Benchmark

This section applies the general development process in the design of an FTCS for a typical industrial system, the electro-mechanical position servo for speed control of large diesel engines. Although this benchmark equipment is relatively simple, it is well suited to highlight important aspects of real applications. The basic requirement for the FTCS is to improve availability so continued operation is possible in case of the most likely faults. The benchmark FTCS is implemented and tested on a laboratory setup comprising the real industrial hardware. This setup facilitates repeated tests.

3.3.1 Introduction to the Benchmark Equipment

The actuator is part of a ship governor used to control the rotational shaft speed of a diesel engine (Blanke and Nielsen (1990)). It regulates the amount of fuel to the engine by controlling the position of a common rod that connects to a valve on each cylinder. The position is controlled by a digital computer by means of a brush-less DC motor that connects to the rod through an epicyclic gear and an arm (see figure 3.9). Power to

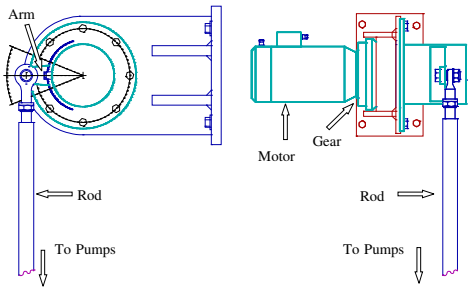


Figure 3.9: *Electro-mechanical actuator for diesel engine speed control.*

the motor is delivered by a switching power drive that has internal current and velocity control. The power drive also has some safety arrangements and monitoring abilities. The position is mechanically limited and two micro switches serve as end-stop detectors. Activation of an end-stop switch inhibits current from the power drive in one direction. An electro-magnetic brake in the motor is disengaged on power-up. It is engaged if power is lost or a primary computer watchdog triggers. The power drive has adjustable limits for peak and mean values of the output current, where the mean value limit is active if the average electrical effect exceeds a certain limit. The motor temperature is

monitored with a thermistor and the current is limited to the mean value if the motor becomes too hot. The electrical-mechanical diagram of the equipment is seen in figure 3.10 and the main characteristics are listed in table 3.4.

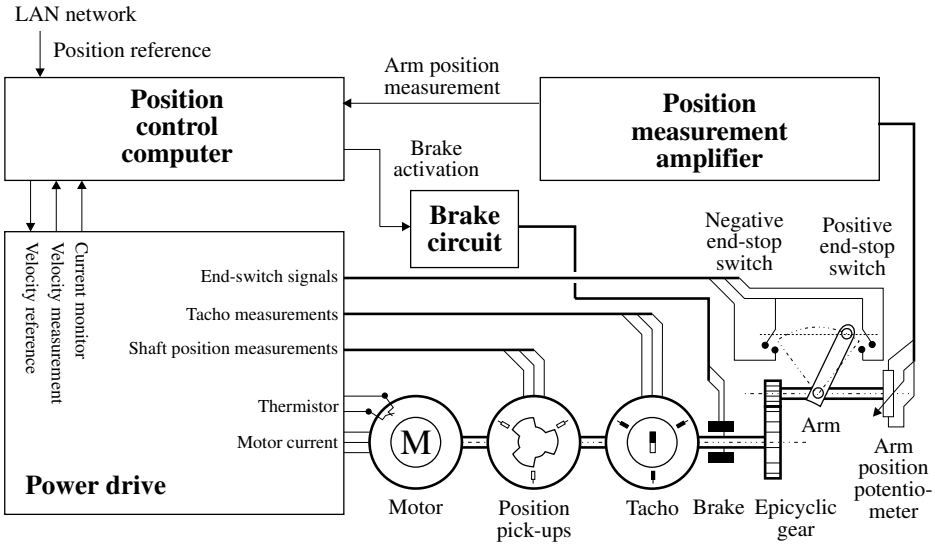


Figure 3.10: *Electrical-mechanical diagram of the benchmark equipment.*

Table 3.4: *Main characteristics of the benchmark actuator.*

Peak power to servo motor	2.5 kW
Peak current	± 30 A
Maximum mean current	± 12 A
Time to full speed (0-3000 RPM)	22 ms
Arm operating range	± 23 degrees
Time for full stroke of arm (0-100%)	270 ms
Velocity control loop time constant	4 - 6 ms[1]
Peak actuator torque on gear output shaft	± 1.2 kNm
Maximum external load torque on gear output shaft	± 0.6 kNm
Computer sampling time	10 ms

[1] *The velocity control loop time constant depends on the load inertia.*

The basic requirement to fault tolerance is that single faults shall be handled, so operation shall continue if possible and otherwise the system shall be closed down with the arm braked at the present position. The faults shall be accommodated before the diesel engine speed is significantly effected, which is translated into a maximum change in arm position of 2 degrees (5% of full range).

3.3.2 System Modelling

Hierarchical system model. The first step in the development procedure is to perform a hierarchical breakdown of the equipment for both the physical structure and the functional structure. The results are presented in figure 3.11. The figure does not show all details, but illustrates the concept.

Fault modelling. The physical structure model forms the basis for an FMEA that lists possible component faults. Table 3.5 shows a representative number of the faults which are considered in this thesis. The actuator is located in a harsh environment so the listed faults can happen due to mechanical vibration, wear, or loose connections in the wiring.

Table 3.5: *Extract of the failure mode and effects analysis for the benchmark equipment.*

Fault No.	Subsystem	Comp./func.	Failure mode	Effect
2.1.1.*1*	Velocity ctrl. electronics	Reference voltage wire	Disconnected	Vvelref=0V
2.1.2.*1*	Velocity ctrl. electronics	Vel. meas. scaling pot.	Disconnected	Vvelmeas=0V
3.2.1.*1*	Motor brake	Input wire	Disconnected	Brake active
4.2.3.*1*	Position potentiometer	Positive wire	Disconnected	Vposmeas=0V
4.2.4.*1*	Position potentiometer	Negative wire	Disconnected	Vposmeas=15V
4.2.5.*1*	Position potentiometer	Output wire	Disconnected	Vposmeas=remains constant
4.3.1.*1*	Positive end-stop switch	Contact	Fails open	Rposswitch=∞
4.3.2.*1*	Negative end-stop switch	Contact	Fails open	Rnegswitch=∞

3.3.3 Fault Propagation Analysis

The second step in the development procedure is to determine the functional discrepancies of the failure effects in table 3.5 and model how these discrepancies propagate through the subsystems in the functional structure model to end-effects on the actual arm position.

The relationships between the fault effects of the physical model to subsystem discrepancies in the functional model are listed in table 3.6. This table maps the outputs of the FMEA in table 3.5 to inputs for the fault propagation analysis.

The FPA is based on a qualitative description of the discrepancies using the notation low, high, zero, and constant. These terms correspond to the variable being lower or higher than is should be, fixed at zero, and fixed at a constant value. The block diagram of the system in figure 3.12 shows the relation between the individual subsystems. Each subsystem is represented by a logic model that describes the behaviour for all combinations of the qualitative values of the inputs. The tables 3.7 through 3.14 are multi-valued truth tables for all the subsystems involved.

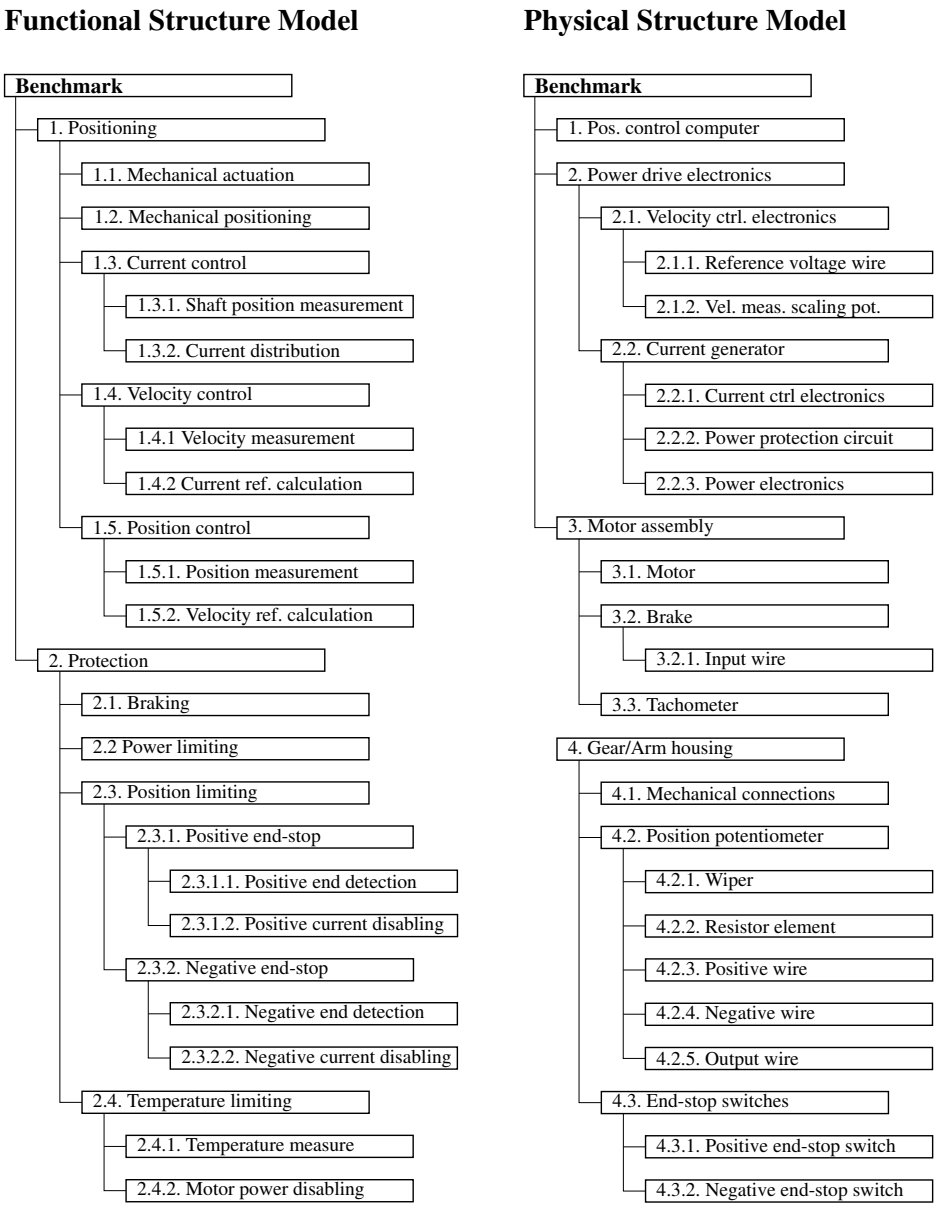


Figure 3.11: *Illustration of two ways to make a hierarchical breakdown of the structure of the benchmark equipment: Functional structure model and Physical structure model.*

Table 3.6: *Relationship between component fault effects and functional discrepancies.*

Fault No.	Fault effect	Subsystem No.	Subsystem	Discrepancy
2.1.1.*1*	Vvelref=0V	1.4.2	Current ref. calculation	VelRefWireDisc
2.1.2.*1*	Vvelmeas=0V	1.4.1	Velocity measurement	TachoWireDisc
3.2.1.*1*	Brake active	2.1	Braking	BrakeFailedOn
4.2.3.*1*	Vposmeas=0V	1.5.1	Position measurement	VposWireDisc
4.2.4.*1*	Vposmeas=15V	1.5.1	Position measurement	VnegWireDisc
4.2.5.*1*	Vposmeas=remains constant	1.5.1	Position measurement	VoutWireDisc
4.3.1.*1*	Rposswitch=∞	2.3.1.1	Positive end detection	EndSwitchPosDisc
4.3.2.*1*	Rnegswitch=∞	2.3.2.1	Negative end detection	EndSwitchNegDisc

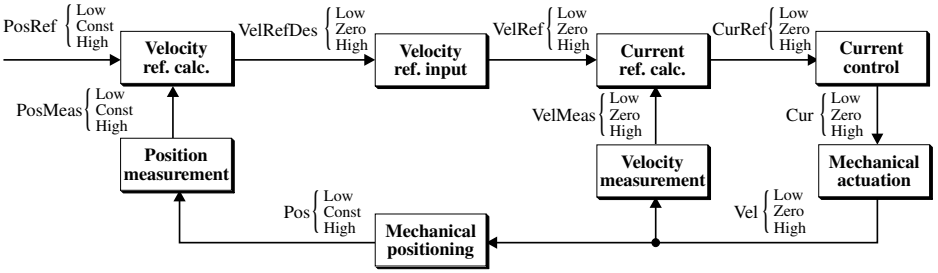


Figure 3.12: *The structure of the fault propagation analysis of the benchmark equipment. A logic description of each block determines the relation between the qualitative inputs and outputs.*

Table 3.7: *Logic model for the position measurement (potentiometer).*

Position measurement	Input	Output
Fault	Pos	PosMeas
NoFault	Low	Low
	Const	Const
	High	High
VnegWireDisc	-	High
VposWireDisc	-	Low
VoutWireDisc	-	Const

Table 3.8: *Logic model for the velocity reference calculation unit (position controller).*

Velocity ref. calc.	Inputs		Output
Fault in loop	PosMeas	PosRef	VelRefDes
No	-	Low	Low
	-	Const	Zero
	-	High	High
Yes	Low	-	High
	high	-	Low
	Const	Low	Low
	Const	High	High

Table 3.9: *Logic model for the velocity reference input (velocity reference wire).*

Velocity ref. input	Input	Output
Fault	VelRefDes	VelRef
NoFault	Low	Low
	Zero	Zero
	High	High
VelRefWireDisc	-	Zero

Table 3.10: *Logic model for the velocity measurement (tachometer).*

Velocity measurement	Input	Output
Fault	Vel	VelMeas
NoFault	Low	Low
	Zero	Zero
	High	High
TachoWireDisc	-	Zero

Table 3.11: *Logic model for the current reference calculation unit (velocity controller).*

Current ref. calc.	Inputs		Output
Fault in loop	VelMeas	VelRef	CurRef
No	-	Low	Low
	-	Zero	Zero
	-	High	High
Yes	Low	-	High
	high	-	Low
	Zero	Low	Low
	Zero	High	High

Table 3.12: *Logic model for the current control unit (power drive).*

Current control	Input	Output
Fault	CurRef	Cur
NoFault	Low	Low
	Zero	Zero
	High	High
EndSwitchPosDisc	High	Zero
EndSwitchNegDisc	Low	Zero

Table 3.13: *Logic model for the mechanical actuation (motor).*

Mechanical actuation	Input	Output
Fault	Cur	Vel
NoFault	Low	Low
	Zero	Zero
	High	High
BrakeFailedOn	-	Zero

Table 3.14: *Logic model for the mechanical positioning (gear and arm).*

Mechanical positioning	Input	Output
Fault	Vel	Pos
NoFault	Low	Low
	Zero	Const
	High	High

The logic models are then connected to a multi-valued graph for further analysis. In this thesis they have been translated to boolean logic rules for Beologic™ (see appendix A.1). The rule base is easily analyzed in this framework for possible faults that would lead to oscillation of the physical system caused by negative feedback loops. This is done, as explained in section 3.2.2, by examining the rule base for bindings on the discrepancy variables representing fault inputs. The discrepancy TachoWireDisc is found to be *bound to false*, which means that this fault may be the origin of an oscillation in the system. The path of propagation of the tacho-fault is shown by the fault tree in figure 3.13, where the physical oscillation is represented as a "cross-over" between two paths of discrepancies. In the real system, a velocity measurement failing zero causes

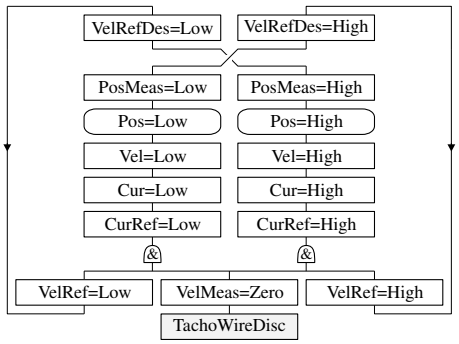


Figure 3.13: *The fault TachoWireDisc causes oscillation of the output arm in the physical benchmark system caused by the sign change in the negative feedback position controller. This property is evident in the corresponding fault tree, shown in this figure, because the path of the fault has a "cross-over" between the "Low"-discrepancies and the "High"-discrepancies.*

the arm to oscillate around the demanded position reference point. A test sequence of this situation from the laboratory setup is seen in figure 3.14. The fault TachoWireDisc is therefore decoupled from the Low-High paths in the FPG by extending all the logic models with an "oscillation" discrepancy. As an example, the new model for the velocity measurement subsystem is given in table 3.15. The "oscillation" discrepancy propagates further to a new end-effect Pos=Oscil that characterises oscillation of the arm position. The Beologic™ rules for the final FPG of the benchmark can be found in appendix A.2.

As a conclusion on the development of the FPG, the complete relationship between faults and end-effects in the benchmark is listed in table 3.16. This table gives an overview of which end-effects have multiple fault reasons and which faults cause more end-effects. Note, that the two end-stop switch faults are modelled as causing a constant position, although movement in one direction is still possible. This is the correct description, because the anomalous situation is present when the arm cannot move.

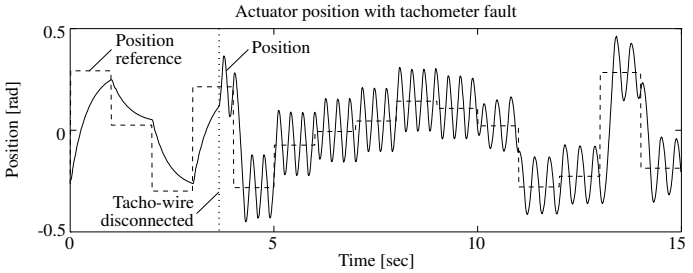


Figure 3.14: *The end-effect of a disconnected tachometer wire is an oscillation of the arm position around the position reference set-point. The fault happens at 3.65 sec.*

Table 3.15: *Modified logic model for the velocity measurement (tachometer), where the effect of the fault TachoWireDisc is changed from zero to a dedicated "oscillation" discrepancy.*

Velocity measurement	Input	Output
Fault	Vel	VelMeas
NoFault	Low	Low
	Zero	Zero
	High	High
	Oscil	**Oscil**
TachoWireDisc	-	**Oscil**

Table 3.16: *The relation between faults and end-effects of the benchmark example.*

↓ Fault - End-effect →	Pos=Low	Pos=Const	Pos=High	Pos=Oscil
VposWireDisc			x	
VnegWireDisc	x			
VoutWireDisc	x		x	
VelRefWireDisc		x		
TachoWireDisc				x
BrakeFailedOn		x		
EndSwitchPosDisc		x		
EndSwitchNegDisc		x		

3.3.4 Severity Assessment

The end-effects, found in the previous step, are classified with respect to severity. The analysis has been continued up to the level of engine speed to determine the top-level consequences. Table 3.17 shows the classification into 5 hazard classes.

Table 3.17: *Severity assessment of end-effects caused by the considered faults in the Benchmark problem.*

	Hazard class	End-effects on ship speed
1	Catastrophic	Acceleration to full speed (Pos=High)
2	Very serious	Deceleration to zero speed (Pos=Low)
3	Serious	Constant speed (Pos=Const)
		Fuel rod position oscillating, possibly ripple on ship speed, extreme wear on equipment (Pos=Oscil)
4	Not serious	
5	Indifferent	

3.3.5 Analysis for Reconfiguration

The next step in the process is to produce a fault tree that shows the causal relation between failure modes and end-effects. The benchmark fault tree shown in figure 3.15 consists of three independent sub-trees.

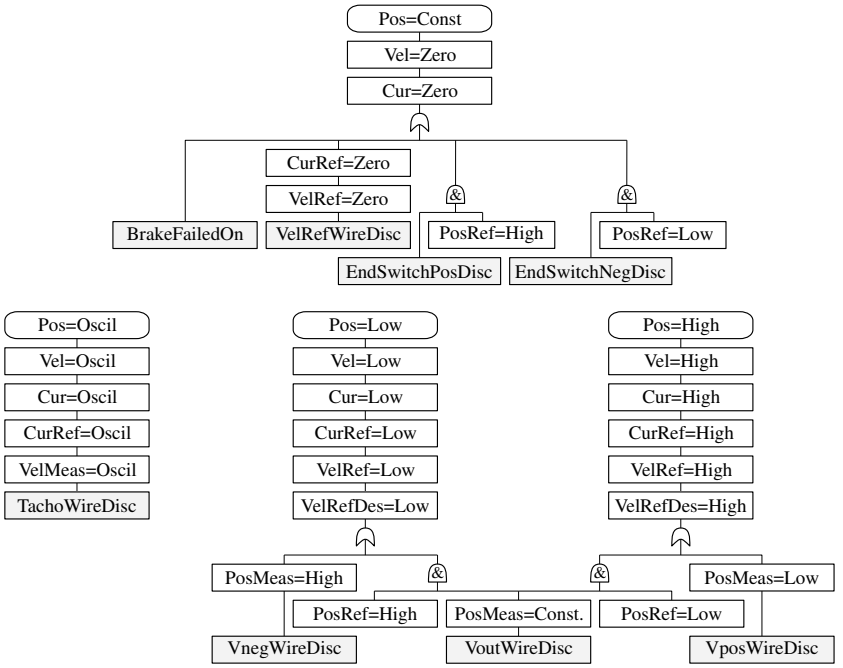


Figure 3.15: *The benchmark fault tree showing the possible faults that cause the four end-effects on the arm position: constant, oscillating, low, and high.*

3.3.6 Remedial Action Selection

The benchmark equipment does not offer many possibilities for reconfiguration, but a few things can be done to maintain operation. The search for remedial actions is supported by an inspection of the fault tree in figure 3.15, where a specific reconfiguration proposal can be examined for its ability to stop fault propagation. Possible remedial actions for the benchmark are listed in table 3.18.

Table 3.18: *Possible remedial actions for the benchmark equipment.*

No.: **Name** Faults	Remedial action	Reconfiguration condition	Degradation
1: **PosEstim** Position measurement faults (PosMeas=High, PosMeas=Low, PosMeas=Const)	Substitute position measurement with filtered integration of velocity measurement. Use softer position control to avoid large velocity changes as these cause estimation errors.	Position error exceeds 5% [1]	Longer position change response times. Limited operation duration due to drift from sensor noise and measurement errors
2 : **VelCtrlBypass** Faults connected to the velocity controller (VelMeas=Oscil, VelRef=Zero)	Disable velocity control loop by switching power drive to current control. Activate alternative position controller that uses the current reference as controllable	Position error exceeds 5% or more than 5 oscillations about position set-point [2]	Longer position change response times
3: **CloseDown** Motor and power drive faults (Cur=Zero, Vel=Zero)	No reconfiguration possibilities on present equipment, so make a close-down (brake) [3]	Position changes more than 5% [4]	System inoperable. Fuel pump rod fixed at last active position

[1] *These faults cause either a change to full or zero engine speed and it is undesirable to have a large actuator position change before reconfiguration is activated. It may also be unsafe to use a position estimate based on velocity, if there has been large velocity changes because the discrete sampling of the velocity causes position estimation errors. A full stroke is 270 ms, so the reaction time must be considerably smaller.*

[2] *The end-effect of a zero velocity measurement is an oscillation about the position set-point, but the amplitude is not yet known. The condition for reconfiguration is either that the position changes more than 5% or that there has been a sufficient number of oscillations to expose the equipment to excess wear.*

[3] *The equipment could be slightly modified, so the control computer could override the end-stop switch signals by a digital output. This would reduce safe operation as the end-stop switches are mounted to protect the equipment, but operation could continue un-effected. Also, the reliability of the system would degrade because additional electronics (wires, relays, etc.) would have to be mounted.*

[4] *The end-stop switch failure still facilitates motion in one direction, but it is impossible to return in the opposite direction. The reconfiguration condition is thus formulated as a position change because this is the undesired effect.*

The variables in brackets in the first column show the points in the fault tree, where fault propagation is stopped by the corresponding reconfiguration. It can be seen that it is possible to prevent the occurrence of any end-effect with one of these remedial actions. It should be noted, whatsoever, that a shut-down also causes the arm to be fixed at the present position (same as the Pos=Const end-effect).

The *reconfiguration condition*, that determines the maximum tolerable position change before the remedial action is executed, is considered for each remedial action. The overall requirement was stated in section 3.3.1 as a maximum position error of 5% corresponding to 2 degrees deviation of the arm position.

The last column in table 3.18 lists the operational degradation following a recon-

figuration. There is no redundant hardware in the equipment, so it is not possible to reconfigure without any degradation. Nevertheless, it is extremely important to have at least some control of the system until other contingency procedures have been effectuated by the operator.

3.3.7 Fault Detection and Isolation Design

The basic requirements to FDI are given in the remedial action list (table 3.18). It can be seen that the following three groups of discrepancies must be detected and isolated:

1. PosMeas=High, PosMeas=Low, and PosMeas=Const
2. VelMeas=Oscil (same as VelMeas=Low and VelMeas=High) and VelRef=Zero
3. Cur=Zero and Vel=Zero

From the *reconfiguration condition* in the same table, it is clear that simple out-of-range detections of the involved variables are not feasible. The zero-discrepancies are within the legal range and the low/high discrepancies must be detected before the position change becomes big enough to exceed the operational range. The detection must then be performed using analytical redundancy where the mathematical models of the velocity controller, the power drive, and the mechanics are utilized.

It is possible to do an initial feasibility study of model based FDI using the FPG, before the heavy job of FDI design is begun. The principle of model based FDI is, basically, to detect deviations between measurable signals. The measurable signals in the benchmark are velocity reference, velocity, current, and position. An example on a fault detector is an observer with velocity and position measurements as inputs that models the gear and the integration from velocity to position. Such an observer is presented in section 4.3 on page 74. This detector is sensitive to position and velocity measurement faults. The logic model of this detector (table 3.19) is then connected to the FPG. Likewise, detectors for deviation between the other measurable signals are connected to the FPG. The Beologic™ rules for these detectors are found in appendix A.3. It is then possible to analyze the combined FPG/detector rule base to determine which faults will be detected by the individual detectors. The benchmarks fault-detector matrix is shown in table 3.20. The faults have been organized into three groups that shall be isolated. From an inspection of the matrix it is obvious that the VelRefWireDisc fault cannot be distinguished from the end-switch faults. It may be possible to isolate VelRefWireDisc if information about the dynamic responses to the involved faults are included in the detector algorithm. This problem is then postponed to the actual FDI algorithm design. It is also obvious that the two detectors, DetVelCur and DetVelRefCur, are superfluous from an isolation point of view. It is possible to isolate the three groups (except the VelRefWireDisc) with the logic relations given in table 3.21, that do not include these two detectors.

Table 3.19: *Logic model of a fault detector that uses the position and velocity measurements for detection of faults in these sensors.*

DetVelPos		
Inputs		Output
PosMeas	VelMeas	DetVelPos
Low	Low	False
Low	Zero	True
Low	High	True
Const	Low	True
Const	Zero	False
Const	High	True
High	Low	True
High	Zero	True
High	High	False
Oscil	Oscil	True

Table 3.20: *A matrix showing the relationship between faults and suggested fault detectors for the benchmark example. A "True" means that the fault can be detected by the corresponding detector. The fault detectors detect inconsistencies between two measurable signals.*

		Fault detectors (named "Det<signal1><signal2>")			
Fault group	Fault	DetVelPos	DetVelRefVel	DetVelCur	DetVelRefCur
PosMeasFault	VposWireDisc	True	False	False	False
	VnegWireDisc	True	False	False	False
	VoutWireDisc	True	False	False	False
VelCtrlFault	TachoWireDisc	True	True	False	True
	VelRefWireDisc	False	True	False	True
OtherFault	BrakeFailedOn	False	True	True	False
	EndSwitchPosDisc	False	True	False	True
	EndSwitchNegDisc	False	True	False	True

Table 3.21: *The benchmark fault isolation Beologic™ rules.*

#	Rule
0	`IsoPosMeasFault = (DetVelPos and not DetVelRefVel)`
1	`IsoVelCtrlFault = (DetVelPos and DetVelRefVel)`
2	`IsoOtherFault = (not DetVelPos and DetVelRefVel)`

This feasibility study of detectability and isolability must be considered as being only preliminary, because the logic models do not include dynamics, disturbances, external load, and unmodelled dynamics - all factors that influence the FDI performance.

The actual design of the two FDI algorithms for DetVelPos and DetVelRefVel are designed at this point in the development procedure, but only DetVelPos is presented in this report. DetVelPos is realized as the observer designed in section 4.3 in the next chapter. This observer is able to detect discrepancies between the position and velocity

measurements corresponding to a position error of 0.3 degrees, which fulfils the reconfiguration requirements of 2 degrees from table 3.18.

3.3.8 Fault Accommodation Design

Fault accommodation in the benchmark case involves the design of the remedial actions listed in table 3.18. The two secondary position controllers and the position estimation algorithm are designed off-line so the run-time handling requires only simple switching between separate configurations of the control level. There is no need to consider the transition between the different configurations.

The basic requirements for the design of the two secondary position controllers were given in table 3.18. The controller for remedial action #1 shall have a lower bandwidth than the primary controller to avoid large changes in velocity, as this causes estimation errors in the velocity based position estimation. The controller for remedial action #2 shall be designed to use the current reference as control output instead of the velocity reference. These requirements are refined for the detailed design of the algorithms. The results are not included in this thesis.

3.3.9 Supervisor Decision Logic Design

The requirements to the benchmark supervisor's decision logic are compiled from the analysis above and summarized in table 3.22, where also commands and monitoring messages to higher levels are included. The additional requirements are not used in this paragraph, but they will be dealt with under the implementation of the supervisor in section 3.4.

The design of the fault handling part of the decision logic is very simple for the present case. The three remedial actions listed in table 3.18 shall be activated/deactivated by the output from the FDI algorithms (table 3.21) in a one-to-one scheme, such that IsoPosMeasFault governs remedial action #1, IsoVelCtrlFault governs remedial action #2, and IsoOtherFault governs remedial action #3. The simple Beologic™ rules for this scheme are listed in table 3.23.

The task is now to verify the decision logic's fault handling functionality in combination with the FPG of section 3.3.3. The logic models are therefore modified to include the functionality of reconfiguration. The two subsystems effected by the reconfiguration are shown in table 3.24 and table 3.25. Note that the two models also include the "oscillation" discrepancy that was added in section 3.3.3. The complete logic model of the system is now available including the FPG, the FDI logic, the decision logic rules, and the reconfiguration (corresponding to the general scheme in figure 3.8). The boolean Beologic™ rules for this model are listed in appendix A.4.

Table 3.22: *Requirements to the benchmark supervisor's decision logic design.*

Input event	Value	Actions
CmdStartStop	Start	Disengage brake, enable power, enable position controller in default primary mode (PosCtrlPrim), enable FDI.
	Stop	Disable power, engage brake.
CmdSmallPerturb	On	Test mode: Generate 5% steps around current position reference.
	Off	Disable small perturbations test.
CmdLargePerturb	On	Test mode: Generate full stroke steps in position
	Off	Disable large perturbations test.
CmdPosCtrlBypass	On	Test mode: Bypass position controller and apply velocity reference directly.
	Off	Restore position control
CmdRestoreNormal		Restore to normal controller configuration. This command is sent by the operator when a detected fault has been manually repaired or if the event was concluded to be a false alarm.
SetPosRef	PosRef	Update position reference to PosRef.
SetVelRef	VelRef	In PosCtrlBypass mode, update velocity reference to VelRef.
IsoPosMeasFault		Switch position acquisition to estimation from velocity measurement, enable position controller in PosCtrlPosEstim mode, disable FDI, send alarm message. Run this mode for 5 minutes then close-down and send alarm message.
IsoVelCtrlFault		Disable velocity controller and use current input instead, enable position control in PosCtrlVelBypass mode, disable FDI, send alarm message.
IsoOtherFault		Disable power, engage brake, send alarm message.
TimPosEstimTimeout		Time-out signal from an external timer used to force close-down some minutes after a position measurement fault (IsoPosMeasFault) caused the position acquisition to be based on the velocity measurement.

Key Cmd : *Command from plant wide control system.*
Set : *Reference set-point from plant wide control system.*
Iso : *Detected and isolated fault event.*
Tim : *External timer event.*

Table 3.23: *The benchmark supervisor's Beologic™ rules for fault handling decision logic.*

#	Rule
0	`PosEstim = IsoPosMeasFault`
1	`VelCtrlBypass = IsoVelCtrlFault`
2	`CloseDown = IsoOtherFault`

Assisted by Beologic™, the complete logic system is analysed for the properties outlined in section 3.2.8. It is not difficult to see, in this simple case, that the decision logic successfully stops the propagation of position and velocity measurement faults and makes a shut-down in the other cases. There is, nevertheless, a number of useful properties that can be observed from the coherence of the complete logic model. The following characteristics of the rule base illustrate the potential of the outlined method:

- An automatic consistency check on the rule base yields the following faults *bound to false*: VposWireDisc, VnegWireDisc, VoutWireDisc, and TachoWireDisc. These faults will thus be reconfigured correctly.

Table 3.24: *Modified model for the position measurement (potentiometer) where reconfiguration is included. The potentiometer faults will only propagate when the potentiometer is in use (PosEstim=Off).*

Position measurement	Inputs		Output
Fault	PosEstim	Pos	PosMeas
NoFault	On	Low	Low
	On	Const	Const
	On	High	High
	On	Oscil	Oscil
VnegWireDisc	Off	-	High
VposWireDisc	Off	-	Low
VoutWireDisc	Off	-	Const

Table 3.25: *Modified model for the velocity reference (velocity reference wire) where reconfiguration is included. If the velocity controller is bypassed (VelCtrlBypass=On) then the output of this unit becomes the current reference instead of the velocity reference and the disconnected velocity reference wire will have no effect.*

Velocity ref. input	Inputs		Output	
Fault	VelCtrlBypass	VelRefDes	VelRef	CurRef
NoFault	Off	Low	Low	-
	Off	Zero	Zero	-
	Off	High	High	-
	Off	Oscil	Oscil	-
VelRefWireDisc	Off	-	Zero	-
-	On	Low	-	Low
	On	Zero	-	Zero
	On	High	-	High
	On	Oscil	-	Oscil

- Likewise, the end-effects are analysed with the following result:
  Pos=Low false
  Pos=Const true
  Pos=High false
  Pos=Oscil false
  This means that the three faults causing the arm to accellerate in either positive or negative direction or start oscillating are handling correctly. The binding on Pos=Const means that all other faults will cause the arm position to freeze, which is correct.

- The binding on Pos=Const is further examined by searching for solutions, and the possible faults are VelRefWireDisc, BrakeFailedOn, EndSwitchPosDisc, and EndSwitchNegDisc. This means that these faults will cause the arm position to freeze. The velocity reference fault (VelRefWireDisc) is included in this list be-

cause it cannot be isolated from the end-switch faults and is handled by a shut-down (see section 3.3.7).

This analysis completes the verification of the supervisor decision logic.

3.3.10 Summary of FTCS design for the Benchmark

The complete the design of a fault tolerant control system for the benchmark actuator has now been presented. Two ways to reconfigure the system have been applied to keep the system running when the position measurement or the velocity controller fails. A scheme for fault detection and isolation was established and decision logic rules were assigned to handle the reconfiguration. The ingredients of the FTCS are then ready for implementation in the three layer structure presented in section 3.1. This is the topic of the last section in this chapter.

3.4 Supervisor Architecture for the Benchmark

The last section in this chapter shows how a supervisor for the benchmark actuator can be implemented in the three layer structure presented in section 3.1. The bottom layer consists of the reconfigurable position control loop, the second layer consists of the fault detectors and effectors, and the third layer consists of the decision logic. The supervisor is implemented to meet the requirements for fault handling and operational commands as specified in the previous section.

3.4.1 Control Level

The position control loop is expanded with the extra functionality for reconfiguration as required by the list in table 3.22. This includes switching between different position control algorithms, changing to position estimation based on velocity measurement, and bypass of the velocity controller in the power drive. The components of this reconfigurable control level is presented in figure 3.16.

The data flow management in the control level has been implemented simply as if-then structures in the controller task. The different configurations of the control level can be tested separately in this arrangement.

3.4.2 Detector-Effector Level

The detector and effector modules are shown in figure 3.17. The two detectors (DetVelPos and DetVelRefVel) use three control level signals to create fault symptoms and the isolation logic (see table 3.21) distinguishes between the two possible fault events: Position measurement fault and velocity controller fault. The fault detectors can be disabled by the decision logic, which is used when a fault has been detected.

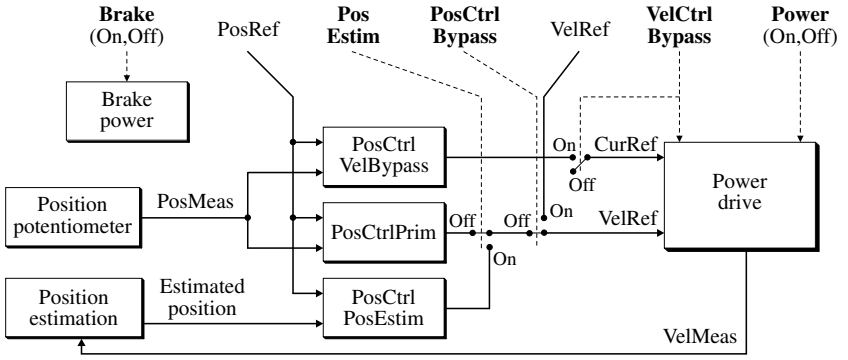


Figure 3.16: *Reconfigurable controller for the benchmark actuator. All the inputs in the top of the figure are settings coming from the effectors and the supervisor decision logic. Boldface variables indicate remedial actions and solid lines are data signals.*

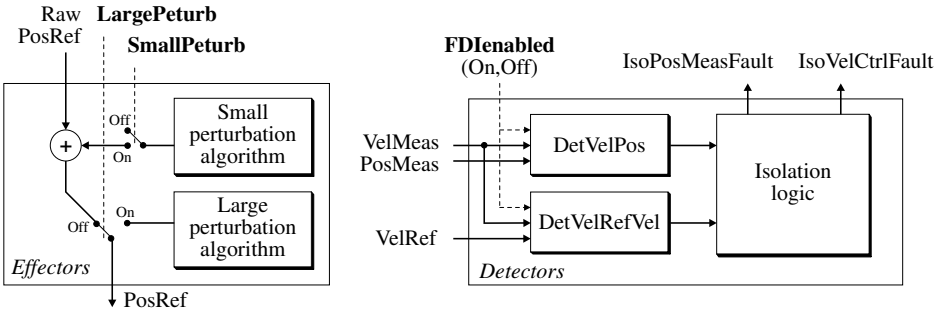


Figure 3.17: *Detector-effector level for the benchmark supervisor. Two fault detectors and isolation logic produce fault events based on input from the control level. The effector box consists of the position reference perturbation generators.*

The effector box consists of two perturbation algorithms used by the operator for test purposes. SmallPerturb adds small steps to the current position reference whereas LargePerturb substitutes the position reference with full stroke steps. There is no specific reconfiguration algorithms for fault handling, because the remedial actions determined in section 3.3.8 are all realised by direct switching in the control level. The two perturbation algorithms are conveniently located in the effector box, because they are independent of the control level and can be assigned to separate real-time tasks.

3.4.3 Decision Logic Level

The decison logic is implemented as a set of state-event machines (SEMs), where the states represent the setting of the different switches in the control level, the effectors and the detectors (see figure 3.18). These SEMs define the complete reaction scheme on input events (commands, set-points, and faults). The difficulty in setting up a SEM scheme is the choice of functionality for each SEM. More SEMs could be joined to one by expanding the state space to all combinations, but this is normally not advantageous. To keep the design simple, it is beneficial to let each SEM be as small as possible and then include dependencies between the individual SEMs as conditions on transitions. This issue is treated in more detail in Zamanabadi *et al.* (1996). All the SEMs in the benchmark decision logic can be seen to have only two possible values, but internal dependencies make the entire SEM suite an involved decision device.

The SEM matrices are mostly self-instructional, but a few comments highlight specific details:

- The requirement to run the position estimation based control for only some minutes has been implemented using an external timer. When the position estimation is started (PosEstim=On) a timer is enabled by the output PosEstimTimerOn. At time-out, the input TimPosEstimTimeout is sent to the decision logic and this forces a close-down.
- An example on dependencies between SEMs is illustrated between SmallPerturb and LargePerturb. The dashed lines indicate dependencies that inhibit simultaneous activation of both perturbation generators.
- A second example on internal dependencies is shown from PosEstim to Brake and Power. When the TimPosEstimtTimeout signal is received, the system is closed down by *forcing* Power to Off and Brake to On.
- All commands to the supervisor are replied with a verification message that is either Failed or Ok. Additional diagnostic information can follow a "VerifFailed" message, but this is not shown in the figure.
- Alarm messages are sent to the plant wide control system when the actuator is autonomously reconfigured. Details on the cause of reconfiguration follows the alarm, but is not included in the figure.

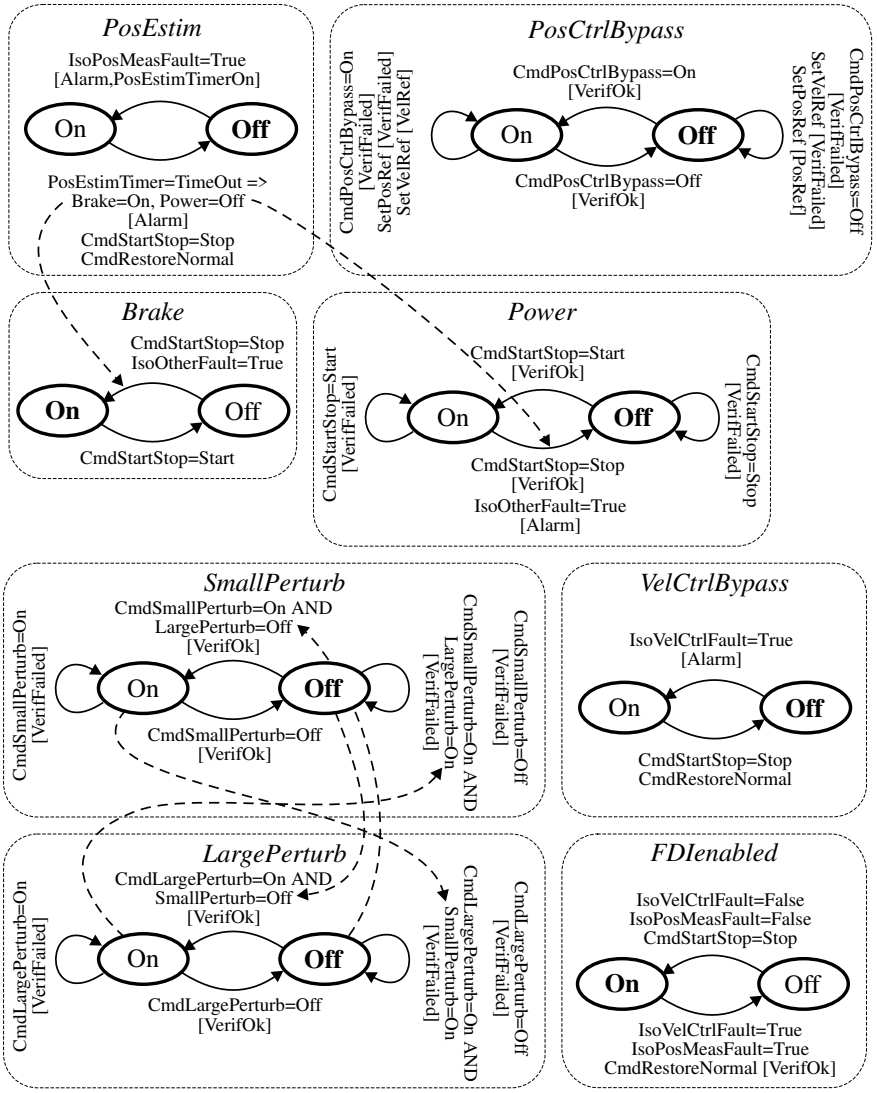


Figure 3.18: *The benchmark supervisor decision logic suite of state-event machines. Default states are in boldface. Outputs are in square brackets. Dashed lines show internal dependencies between SEMs.*

The decision logic is now tested and verified for correct operation. This verification is different from the completeness check described in section 3.3.9, because it concerns the *sequential* behaviour of the state transitions whereas the completeness check is a *single parse* analysis. The checks described below involve only the operation of the SEMs and does not (as for the completeness check) check up against the logic model of the system described by the FPG in section 3.3.3. The characteristics, that are desired to verify, are the following:

- *Dead-ends*. The decision logic can be examined for SEMs that can be locked in a state where there is no exit to other states.
- *Superfluous information*. The SEMs can be analyzed for states that can never be entered and also for superfluous inputs, outputs, and rules. This excessive information represents potential errors in the design.
- *Contradiction*. The SEM rule base can be examined for contradictions between rules. A contradiction exists, if two rules try to force one SEM into two different states.
- *Forbidden state combinations*. The supervisor requirements indicate which combinations of states are not allowed. The final decision logic must be checked up against a list of these forbidden combinations. An example in the benchmark case is SmallPerturb and LargePerturb that are not allowed to be both On. The forbidden combinations can be found by temporarily including transition rules that violate the forbidden relations. When a contradiction check is performed, as explained under the previous bullet, these additional rules will be reported as being in contradiction with the existing SEM rules.

These apects have been verified for the benchmark FTCS using the Beologic VisualState™ tool-box (Beologic (1996)). The Beologic™ AIT tool-box used earlier for the FPG analysis is not designed for SEM analysis as is VisualState™. VisualState™ is, on the other hand, not able to perform a logical verification as AIT, so it is necessary to use both software packages in a two-step verification. VisualState™ is therefore used to simulate the operation of the decision logic and perform the above checks. The software tool facilitates automatic code generation, so the final decision logic module can be automatically generated and compiled into the software on the benchmark control computer.

3.4.4 Experiments in the Laboratory

The performance of the implemented supervisory control system is verified through experiments on a laboratory setup. The laboratory setup is equipped with the real power drive, actuator motor, gear, arm, and rod. It facilitates a programmable load force on the rod generated by a motor arrangement similar to the actuator part.

An experiment is presented in figure 3.19 where the actuator position reference is changed stepwise within 75% of the full range. A random load torque is added as a pseudo random binary sequence (PRBS) with 30% of maximum value. A step-like fault is induced by disconnecting the negative wire of the position potentiometer at 5.42 sec. This causes the position measurement to jump to 0.51 rad, which is detected in the next sample by a first order FDI observer that monitors velocity and position. Subsequently, the faulty position measurement is substituted with an estimate based on the velocity measurement.

The figure shows the position signals with and without FDIR. When FDIR is not

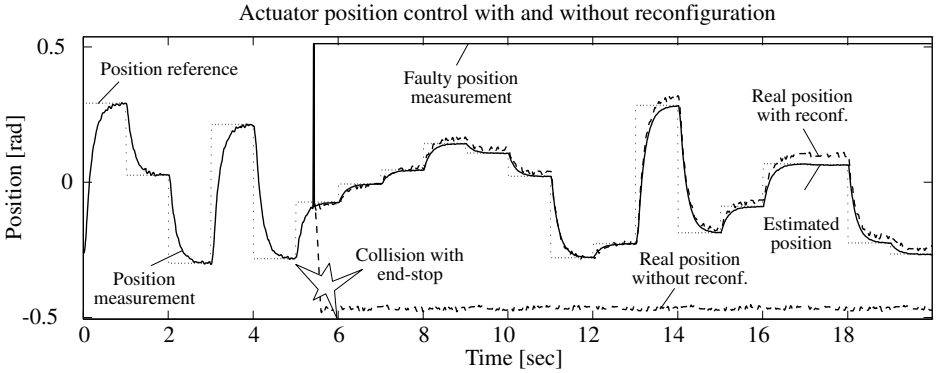


Figure 3.19: *An experiment with the actuator benchmark illustrating the reaction to a position measurement fault with and without fault detection and reconfiguration.*

applied, the position controller forces the arm to the negative end-stop in only 180 msec with the very serious consequence that the diesel engine speed is reduced to zero. When FDIR is applied, an alarm is issued to the operator and operation continues on the estimated position. It is clear from the figure that noise and disturbances cause a bias on the position estimate. After 10 sec this bias is about 0.03 rad (8% of full range). There has been taken no means to estimate this bias prior to the fault, but it is feasible to improve performance with the position estimate by compensating the bias. The experiment illustrates the situation, where the operation can be continued a few minutes until other contingency procedures have been organized.

3.5 Summary

The realization of an FTCS contains basically three tasks in addition to the existing control function: Fault detection and isolation, decision taking, and reconfiguration. A consistent development methodology, presented in this chapter, enables the design engineer to include fault analysis in the early controller design phase and establish specifications

for the three FTCS elements. The preliminary fault analysis is based on a systematic examination of potential component failures and an analysis of the fault effect propagation through subsystems in a simple logic-value model of the system. Assisted by software tools, the preliminary fault analysis is used to determine the end-effect on system level. This is particularly difficult when internal feedback loops are present in the system, and a method to handle this was given. The preliminary analysis supports the specification of requirements for fault handling, and thereby enables early design of the decision logic responsible for fault handling. Combined with the logic-value fault propagation model, this decision logic design can be verified for complete fault coverage and consistency. A procedure for this verification, that has a potential for automation in software, was provided in this chapter.

Application of the development methodology to the benchmark equipment illustrated the potentials of the method. The case study showed how feedback in a cascaded controller structure, that leads to oscillation in case of a tachometer failure, is handled. It also demonstrated a method to analyse the possibilities for isolation between more faults in the same subsystem. Finally, it illustrated the realization of a supervisor's decision logic as state-event machines and how it can be checked for consistency.

Chapter 4

Fault Detection and Isolation on the Benchmark

The general fault detection and isolation problem was introduced in section 3.2.6 and state-of-the-art was presented in section 2.3.2. This chapter compares a wide range of FDI algorithms that have been applied to the benchmark problem. The benchmark proposal was motivated by the scarcity of real industrial test platforms for comparison of advanced FDI methods. Since the announcement in 1993, nine papers have been published with various approaches to solve the problem. This chapter gives a short introduction to the individual methods and discusses the pros and cons in connection with the benchmark problem.

4.1 The Benchmark Proposal

The diesel engine actuator introduced in section 3.3.1 was proposed for an international benchmark test by the Department of Control Engineering at Aalborg University at the Tooldiag'93 conference in Toulouse, France. The purpose was to establish a common platform for comparison between miscellaneous approaches for analytical fault detection and isolation based on a real application. A benchmark kit was distributed with mathematical models of the system implemented in Matlab Simulink™, data sequences with fault scenarios, and design specifications for FDI (Nielsen *et al.* (1993)). The response was very positive and six groups presented their results at the Safeprocess'94 symposium in Helsinki, Finland. These papers were later published in revised form in the Control Engineering Practice journal. Three additional papers have subsequently been published.

The benchmark proposal is available in Blanke and Patton (1995) and a test description is provided in Blanke *et al.* (1995) with all relevant details for FDI studies. The

background for the approaches discussed in this chapter is summarized in this section and the necessary information on the mathematical models is provided in the next section.

Two realistic fault scenarios are considered:

- A temporary interruption between the wiper and the resistance element in the position potentiometer (same as VoutWireDisc in table 3.6 in the previous chapter). This causes the position measurement to remain constant until the fault disappears again.
- A broken wire or defect in the negative end-stop switch (EndSwitchNegDisc in table 3.6 in the previous chapter). This causes the power drive to inhibit negative currents in the motor.

Both faults could cause over-speed of the diesel engine, so computerized fault detection and reconfiguration are required. The available signals for FDI are velocity reference, velocity measurement, and position measurement. The basic requirements to the FDI design are the following:

- The detection delay shall be very short, preferably within one samples.
- The false alarm rate shall be very low, preferably zero.
- The unknown load torque input shall not cause false alarm. No assumptions can be made on the signal except that the maximal load torque on the actuator arm is 0.6 kNm (see table 3.4).
- The FDI scheme shall be robust to model uncertainty. The critical parameters are inertia and friction because they depend on the attached load equipment.

Simulations of the behaviour with the two fault events are provided, where also a load disturbance is included. Five different data-sets are generated with different models, different excitation levels on the velocity reference input, and different periods where the position fault, current fault and load disturbances are enabled. An overview of the data-sets is given in table 4.1. The linear and the simplified nonlinear models are used for linear and nonlinear design approaches, whereas the full scale simulations are used for verification of the design. The full scale nonlinear model is a very close approximation to the real equipment. The high excitation simulation DS4 includes periods with current saturation and can be used to examine robustness properties to unmodelled nonlinearities. DS5 involves a long period with current fault and also simultaneous load disturbances. It can be used to verify isolability between the current fault and the load disturbance. Plots of all data-sets except DS2 are presented in figure 4.1 – 4.4.

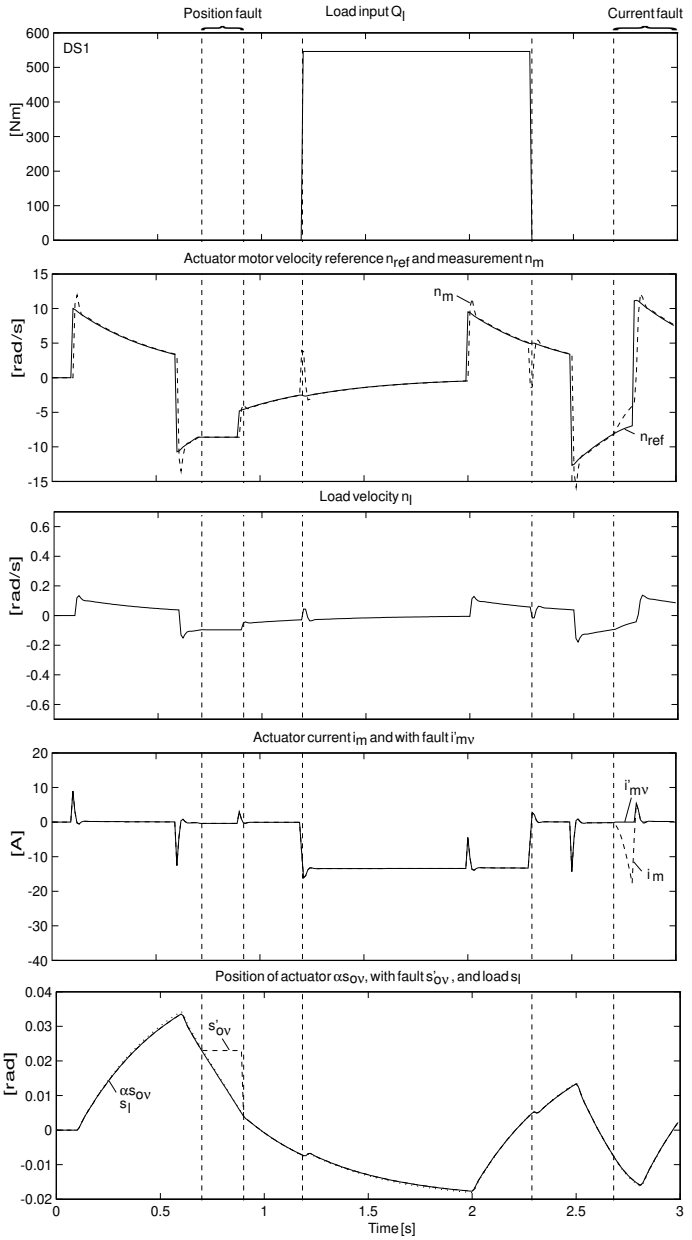


Figure 4.1: *Benchmark test data sequence DS1: Small excitation of linear design model. .*

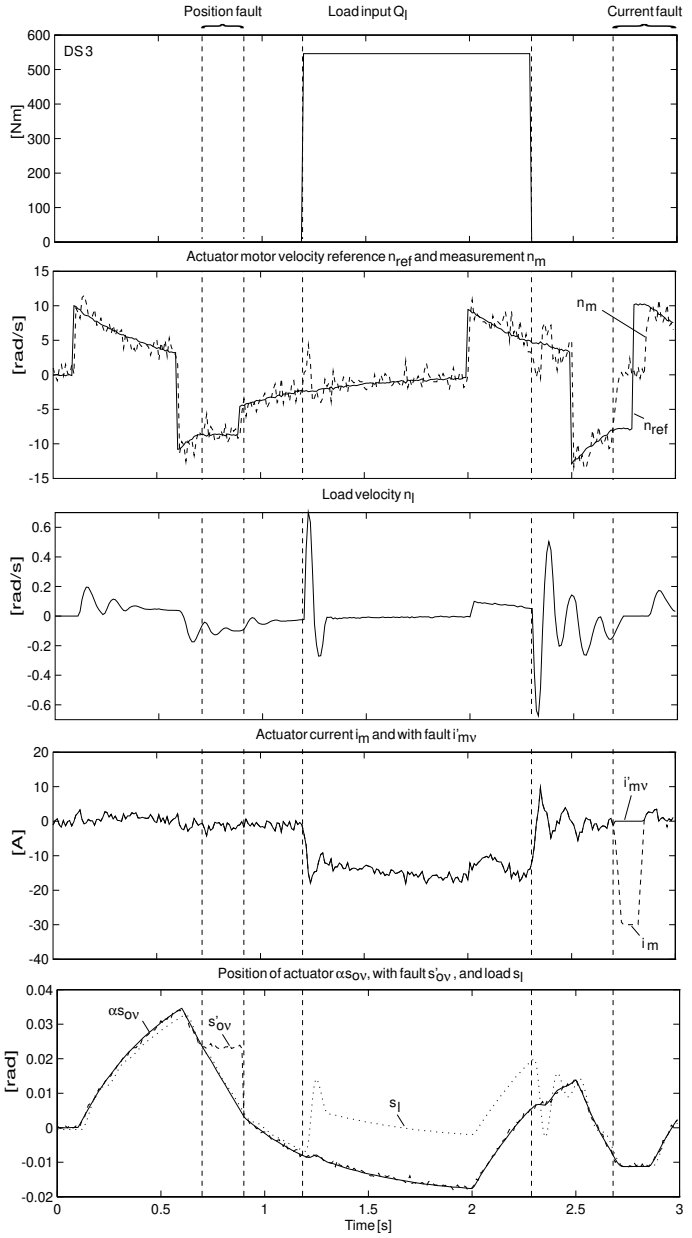


Figure 4.2: *Benchmark test data sequence DS3: Small excitation of full scale nonlinear model. .*

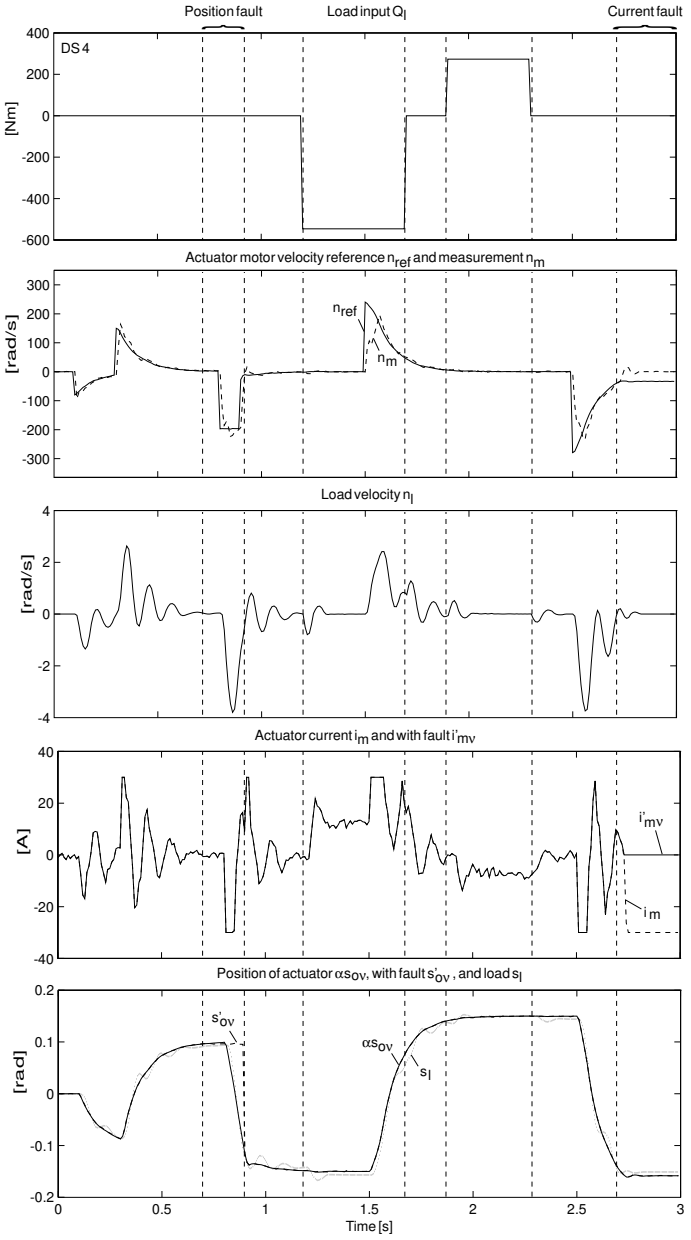


Figure 4.3: *Benchmark test data sequence DS4: High excitation of full scale nonlinear model. .*

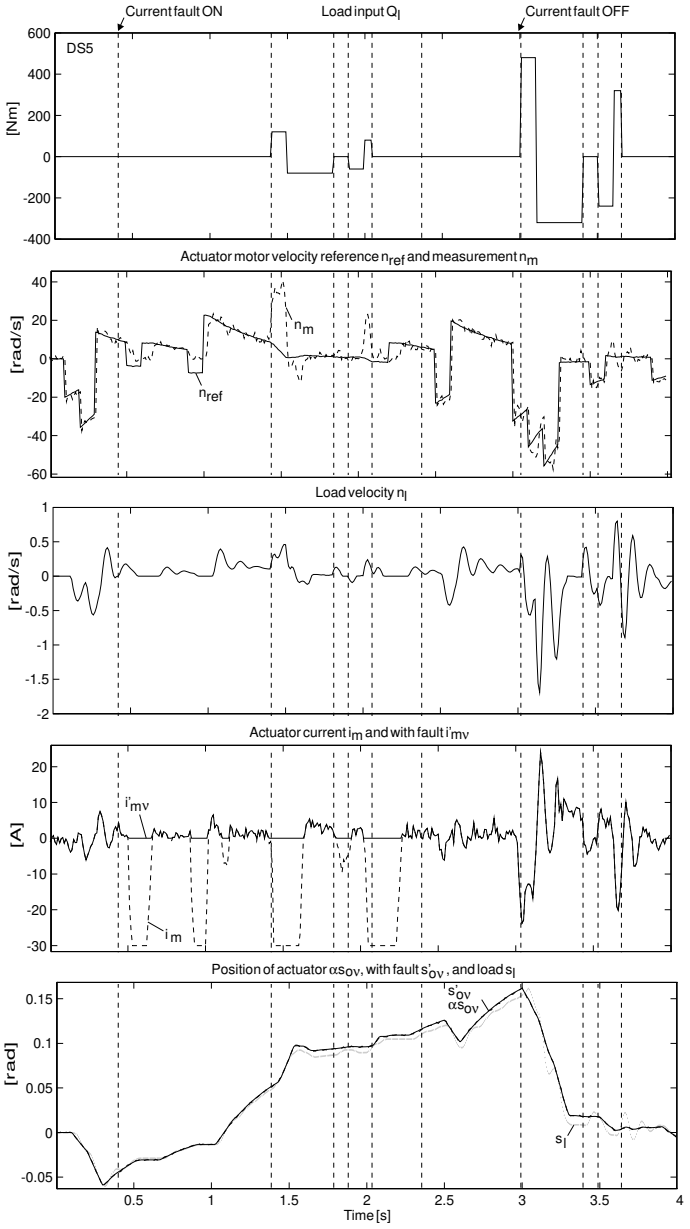


Figure 4.4: *Benchmark test data sequence DS5: Medium excitation of full scale nonlinear model with long current fault period. .*

Table 4.1: *Overview of the data-sets used in the benchmark test. The table describes for each data-set which model was used for generation, the excitation level on the reference input, and the periods were the faults and load inputs are enabled.*

Name	Model type	Excitat.	Pos. fault [msec]	Cur. fault [msec]	Load [msec]
DS1	Linear design model	Low	70-90	270-300	120-230
DS2	Simplified nonlinear model	Low	70-90	270-300	120-230
DS3	Full scale nonlinear model	Low	70-90	270-300	120-230
DS4	Full scale nonlinear model	High	70-90	270-300	120-170 190-230
DS5	Full scale nonlinear model	Medium	-	40-230	140-180 190-205 300-340 350-365

The different FDI approaches presented in this chapter are compared on characteristics like detection delay, false alarm rate, isolability, and the quality of the residuals. The detection delay is measured from the last sample before the faults manifest themselves in the velocity, current, or position signals (see table 4.2).

Table 4.2: *This table lists the time of the first sample, where the benchmark faults manifest themselves in the data signals for the 5 test data-sets.*

Fault	DS1 [msec]	DS2 [msec]	DS3 [msec]	DS4 [msec]	DS5 [msec]
Position fault	70	70	70	74	-
Current fault	270	270	270	273	50

Further details on the benchmark proposal and fault scenarios are available in Blanke *et al.* (1995).

4.2 Benchmark Actuator Description

This section gives a brief introduction to the mathematical model of the benchmark equipment. The necessary information for FDI design is provided in Blanke *et al.* (1995) and further details on the full scale nonlinear model are available in the technical report Bøgh *et al.* (1993).

The fundamental model, shown in figure 4.5, consists of a velocity PI controller within the power drive, an actuator motor, a gear, and an output arm. The velocity reference n_{ref} is generated by a digital position controller. The velocity servo determines the desired motor current i_{m0} to control the motor shaft velocity n_m. Current saturation within the power drive limits the actual current to i_m. The position of the output arm is denoted s_o. The unknown load torque Q_l acts on the motor shaft with a torque Q_{lm}. The figure also includes measurement disturbances and the input location of the two faults

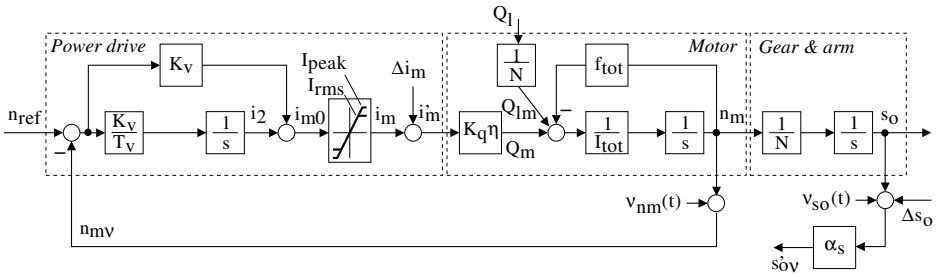


Figure 4.5: *Block diagram of the actuator servo-motor with velocity control loop of the power drive. The speed reference is generated by a digital position controller. The points of additive faults and disturbances are included.*

considered. The parameter values and variables ranges are listed in table 4.3 and table 4.4.

The linear design model provided for the benchmark test is given as the continuous-time state-space model

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}_c\mathbf{x}(t) + \mathbf{B}_c u(t) + \mathbf{E}_c d(t) + \mathbf{F}_{ac} f_a(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{F}_s f_s(t) + \nu(t), \end{aligned} \tag{4.1}$$

where the state vector is $\mathbf{x} = [i_2 \ n_m \ s_o]^T$, the measurable output is $\mathbf{y} = [n_{m\nu} \ s'_{o\nu}]^T$, the control input is $u = n_{ref}$, the load disturbance is $d = Q_l$, the additive actuator fault is $f_a = \Delta i_m$, the additive sensor fault is $f_s = \Delta s_o$, and $\nu = [\nu_{nm} \ \nu_{so}]^T$ are white noise inputs on the velocity and position sensors with corresponding variances K_{nm}^2 and K_{so}^2. The system matrices are

$$\mathbf{A}_c = \begin{bmatrix} 0 & -\frac{K_v}{T_v} & 0 \\ \frac{K_q\eta}{I_{tot}} & \frac{-f_{tot}-K_vK_q\eta}{I_{tot}} & 0 \\ 0 & \frac{1}{N} & 0 \end{bmatrix} \mathbf{B}_c = \begin{bmatrix} \frac{K_v}{T_v} \\ \frac{K_vK_q\eta}{I_{tot}} \\ 0 \end{bmatrix} \mathbf{E}_c = \begin{bmatrix} 0 \\ \frac{1}{NI_{tot}} \\ 0 \end{bmatrix} \tag{4.2}$$

$$\mathbf{F}_{ac} = \begin{bmatrix} 0 \\ \frac{K_q\eta}{I_{tot}} \\ 0 \end{bmatrix} \mathbf{C} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & \alpha_s \end{bmatrix} \mathbf{F}_s = \begin{bmatrix} 0 \\ \alpha_s \end{bmatrix}$$

This model has the velocity feedback noise input ν_{nm} located outside the velocity control loop, which is incorrect corresponding to the real system. The incorrect model was by mistake provided for the benchmark test, so it is used throughout this thesis. The significance of the error is anyway very small.

The corresponding discrete-time representation is

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}\mathbf{x}(k) + \mathbf{B}u(k) + \mathbf{E}d(k) + \mathbf{F}_a f_a(k) \\ \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k) + \mathbf{F}_s f_s(k) + \nu(k), \end{aligned} \tag{4.3}$$

Table 4.3: *Benchmark actuator linear design model parameters.*

Par.	Value	Unit	Description
K_v	0.9	As/rad	Velocity controller gain
T_v	$8.8 \cdot 10^{-3}$	s	Velocity controller integral time
I_{peak}	30	A	Power drive peak current limit
I_{rms}	12.2	A	Power drive mean current limit
f_{tot}	$19.7 \cdot 10^{-3}$	Nms/rad	Total friction referred to servo motor
I_{tot}	$2.53 \cdot 10^{-3}$	kgm^2	Total inertia referred to servo motor
K_q	0.54	Nm/A	Torque constant for servo motor
N	89		Gear ratio
α_s	0.978 (DS1) 1.045 (DS2-DS5)		Measurement scaling factor
η	0.85		Gear efficiency
K_{nm}	1		Noise amplitude on velocity measurement
K_{so}	$5 \cdot 10^{-4}$		Noise amplitude on position measurement

Table 4.4: *Benchmark actuator variables.*

Var.	Range	Unit	Description
i_m	-30 to 30	A	Motor current from power drive
n_m	-314 to 314	rad/s	Shaft speed of servo motor
n_{ref}	-314 to 314	rad/s	Shaft speed reference
Q_{lm}	-6 to 6	Nm	Load torque referred to servo motor
Q_m	-14 to 14	Nm	Load torque referred to servo motor
s_o	-0.4 to 0.4	rad	Output arm position

where the discrete-time system matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{E}$, and $\mathbf{F}_a$ are found for a sampling time of 10 ms.

A full scale 7th order nonlinear description is also provided that includes the following details:

1. Velocity reference input filter in the power drive.
2. Limit on the integral term in the velocity controller (60 A).
3. Peak and mean current saturation in the power drive. The mean limit is active if the energy $i_m^2 t$ exceeds a certain threshold.
4. Ripple on motor torque depending on motor shaft position.
5. Nonlinear friction terms: Stiction, columbic friction, and viscous friction.
6. Small backlash in gear.
7. Small spring effect in the rod, although not intended by design.
8. Analog-digital quantization effects.

The load is modelled in the full scale description as a separate subsystem with inertia and friction connected to the motor subsystem through a spring (rod). The model is presented as nonlinear differential equations that also include hard nonlinearities where the time derivative is discontinuous (Bøgh *et al.* (1993)).

4.3 Approaches to Fault Detection and Isolation on the Benchmark

This section gives a description of the nine methods used in the design of FDI schemes for the actuator position measurement and current faults. Each approach is presented with an outline of the technique, choice of parameters, plots of the decision functions (residuals), and resulting detection delays. The comments and conclusions included in the individual descriptions originate from the associated paper unless specifically stated.

The two faults comprise different challenges for FDI. The current fault is not difficult to detect, but very difficult to isolate from the unknown load disturbance and current saturation. The problem is that the two inputs, Δi_m and Q_l, act at the same point in the system, and cannot be distinguished when they are just considered as additive inputs unless further information (e.g. amplitude or frequency) is included. This makes it difficult because no assumptions can be taken on the load input except a maximum amplitude.

The position fault can be detected by simple means using solely the velocity and position measurements when assuming a fault-free velocity measurement. The following first order observer is considered as a *base-line observer* where the estimation error is used to detect position measurement faults:

$$\begin{aligned} \dot{x} &= Lx + \begin{bmatrix} \frac{1}{N} & \frac{-L}{\alpha_s} \end{bmatrix} \begin{bmatrix} n_{m\nu} \\ s'_{o\nu} \end{bmatrix} \\ r_s &= \frac{1}{\alpha_s} s'_{o\nu} - x. \end{aligned} \tag{4.4}$$

The gain L is selected to give an appropriate trade-off between noise/disturbance attenuation and detectability of incipient position measurement faults. A proper choice is a pole in L=-10 rad/s which leads to the residuals for data-set DS1-DS4 seen in figure 4.6. The major challenge for this detector is the low sampling rate. The actual velocity and position can change between two samples because the velocity control loop time constant (4-6 msec) is much lower than the sampling time (10 msec). This is most evident from the residual plot of DS4, that clearly necessitates a higher threshold than the low excitation residuals in DS1-DS3.

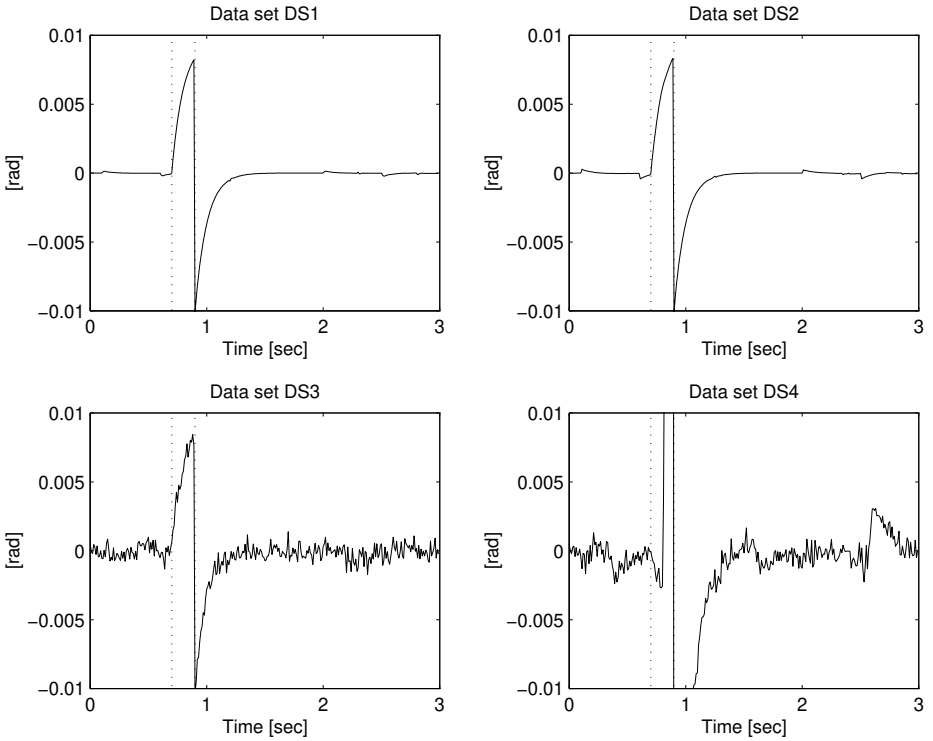


Figure 4.6: *Residuals for position measurement fault detection generated by the base-line observer. Dotted lines indicate the period where the position fault is enabled.*

4.3.1 Observer and Signal Processing Approach

The paper by Höfling *et al.* (1995) presents two different approaches for the detection of position measurement faults and current faults. The former is based on an observer similar to the base-line observer described above and with an adaptive threshold. The latter is based on a dedicated filtering technique.

The position measurement fault is first considered in the context of a systematic parity space design, where all available measurements are used to generate a single residual signal:

$$r_s(k) = \mathbf{v}^T \left(\mathbf{Y}(k) - \mathbf{T}\mathbf{x}(k) - \mathbf{Q}_B \mathbf{U}_{ref}(k) - \mathbf{Q}_E \mathbf{D}_L(k) \right), \tag{4.5}$$

where $\mathbf{v}$ contains the design parameters, $\mathbf{Y}$ holds present and past measurables ($n_{m\nu}$ and $s'_{o\nu}$), $\mathbf{x}$ is the state vector, $\mathbf{U}_{ref}$ is present and past values of n_{ref}, $\mathbf{D}_L$ is present

and past values of Q_l, and the matrices $\mathbf{T}$, $\mathbf{Q}_B$, and $\mathbf{Q}_E$ are given by the system matrices ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{E}$).

The values of the elements in $\mathbf{v}$ are determined to minimize the residual's sensitivity to the load (Q_l), the internal states ($\mathbf{x}$), and $n_{ref}(k)$:

$$\begin{aligned} r_s(k) = {} & 4.59 \cdot 10^{-5} n_{m\nu}(k-1) + 8.31 \cdot 10^{-5} n_{m\nu}(k) \\ & + 1.95 \cdot 10^{-5} n_{ref}(k-1) + s'_{o\nu}(k). \end{aligned} \tag{4.6}$$

The resulting residual is found to be inappropriate for detection because it is too sensitive to noise, load, and unmodelled dynamics compared to the fault sensitivity. The underlying problem is twofold: First, the residual depends on n_{ref}, which is superfluous information, that causes the residual to be dependent on the load and nonlinearities in the velocity controller. Second, there is no filtering to attenuate measurement noise. This example shows the difficulties by applying a standard method without considering the system structure. Parity space design is also considered in section 4.3.8 in connection with current fault detection.

Höfling *et al.* (1995) then recognizes that the position fault is detectable by using only the velocity and position measurements. It is demonstrated that a direct comparison between position and the integrated velocity

$$r_s = s'_{o\nu} - \frac{\alpha_s}{N} \int_0^t n_{m\nu}(\tau) d\tau, \tag{4.7}$$

cannot be used, because the residual drifts due to noise and inter-sampling dynamics

Instead, the base-line observer, presented above, is considered in connection with an adaptive threshold, that takes an "integration-factor-uncertainty", ΔK, into account. The principle is to make an adaptive threshold l_{ad} dependent on velocity:

$$l_{ad} = \left\| \frac{\Delta K}{s + \alpha_s L} n_{m\nu} \right\| + 4\sigma_{resid}, \tag{4.8}$$

where L is the observer gain. A lower limit is chosen to four times the root-mean-square of the fault-free residual, which gives a confidence probability for no false alarms of at least 93.75%. ΔK is selected to 4% of α_s/N, i.e.$4.5 \cdot 10^{-4}$. An observer pole in L=-12 rad/s (almost equal to the base-line design with L=-10 rad/s) is found to be appropriate. The residuals and the adaptive thresholds are shown for different data-sets in figure 4.7. The corresponding detection delays are listed in table 4.5, including a comparison to a fixed threshold test against a threshold of 0.005 rad. The design fulfills the requirement for no false alarms for all data-sets.

The detection of current faults is achieved by applying a signal processing procedure, where a residual is generated by filtering the velocity control error through a fourth order low-pass filter. The residual is then compared to an adaptive threshold that is high during large position reference changes and is downward bounded by a fixed threshold.

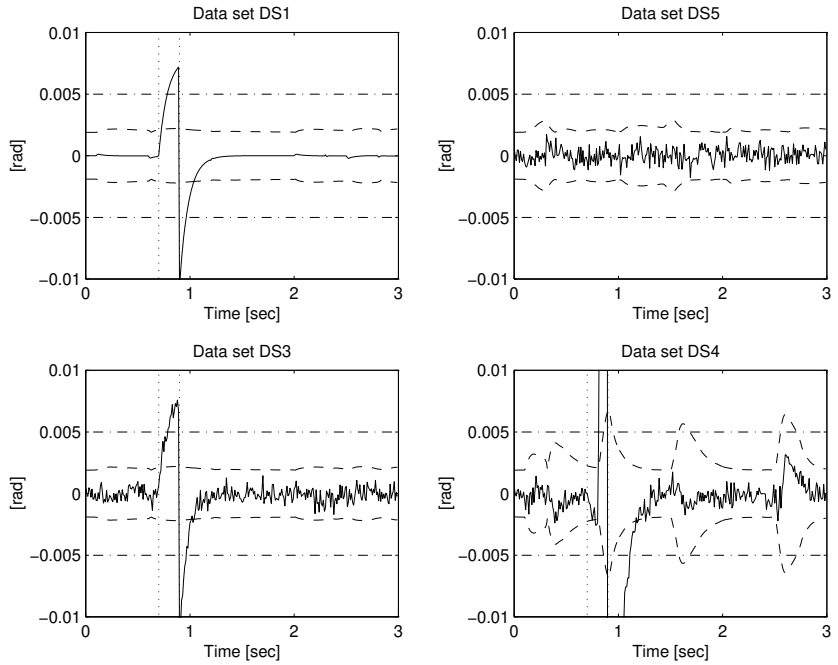


Figure 4.7: *Residuals (solid) with adaptive thresholds (dashed) and fixed thresholds (dash-dotted) for the position fault detection algorithm designed by Höfling* et al. *(1995). Position fault period is indicated with dotted lines.*

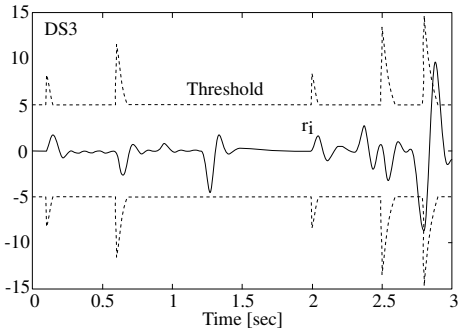


Figure 4.8: *Residuals for data-set DS3 with adaptive thresholds for the current fault detection algorithm designed by Höfling* et al. *(1995).*

Table 4.5: *Fault detection delays for the position fault detection algorithm designed by Höfling* et al. *(1995).*

Detection delay [msec]	DS1	DS2	DS3	DS4
Position fault, observer with adaptive threshold	30	30	30	40
Position fault, observer with fixed threshold	90	80	90	100

An example on this is given in figure 4.8. Different values of the design parameters are chosen for the different data-sets. The detection scheme successfully catches the current faults in all sequences, but false alarms are present in DS5 where the a step in the load is the reason for the current fault to appear. No detection delay values are given in the paper.

4.3.2 Frequency Domain Approach

A frequency domain approach for detection of both position measurement and current faults is presented in García *et al.* (1995). The method uses $\mathcal{H}_\infty$-optimization for the design of an observer that decouples the two faults.

The basic idea is stated in the paper: "Through a frequency characterization of all achievable residual dynamics, based on coprime stable factorizations and an analog result of Youla parameterization to observers, the robust residual generation design is formulated as an optimization problem and solved by $\mathcal{H}_\infty$-optimization techniques"

A residual is generated as a comparison between measured and estimated output $\mathbf{y}(t)$. In generalized format this is given by

$$\mathbf{r}(s) = \mathbf{R}(s)(\hat{\mathbf{M}}_u(s)\mathbf{y}(s) - \hat{\mathbf{N}}_u\mathbf{u}(s)), \tag{4.9}$$

where the transfer matrices $\hat{\mathbf{M}}_u$ and $\hat{\mathbf{N}}_u$ are stable coprime factorizations of the input matrix transfer function $\mathbf{G}_u$ which is given by the system matrices $\mathbf{A}_c, \mathbf{B}_c,$ and $\mathbf{C}$. A procedure is provided in the paper to determine the filter $\mathbf{R}$ so robustness and fault isolation are achieved.

The resulting residual generators are found to be of third order:

$$r_a(s) = \frac{45s^3 + 7707s^2 + 836228s}{(s+100)^3} n_{m\nu}(s) + \frac{7346s^2 + 826228s}{(s+100)^3} n_{ref}(s), \tag{4.10}$$

$$r_s(s) = \frac{0.0106s^2 + 1.8s + 196.7}{(s+100)^3} n_{m\nu}(s) - \frac{s^3 + 171s^2 + 18554s}{(s+100)^3} s'_{o\nu}(s). \tag{4.11}$$

The residuals for the three data-sets, DS2, DS3, and DS4 are shown in figure 4.9. It is clear that it is possible to find a fixed threshold for the position fault detection. It is impossible to use fixed threshold testing for the current fault detection because the residual is too sensitive to external load disturbances and large velocity reference changes.

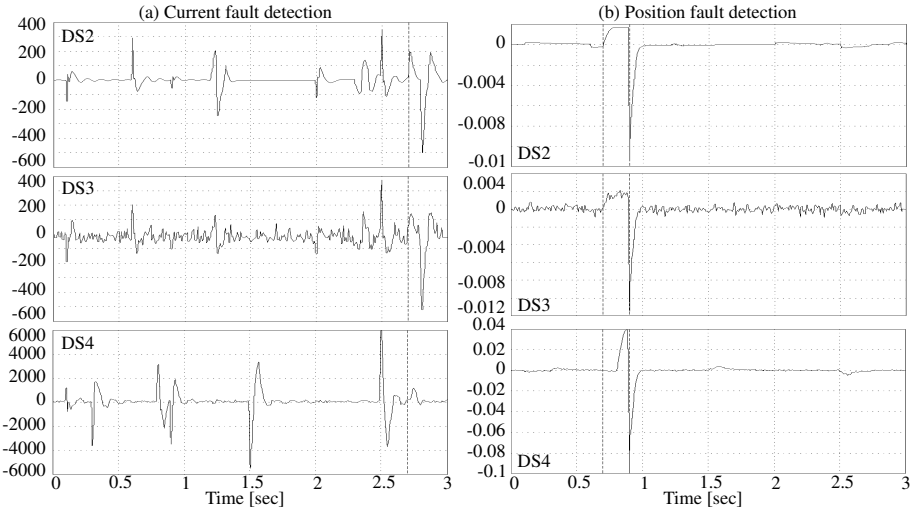


Figure 4.9: *(a) Current fault detection residuals (r_a) and (b) position fault detection residuals (r_s) for the FDI algorithm designed by García* et al. *(1995). Dashed lines indicate fault periods.*

García *et al.* (1995) provides an equation for an adaptive threshold, determined in the frequency domain, but it has not been applied to the benchmark test because no bounds on the unknown load torque was provided in the benchmark proposal. The authors did not analyse the test sequences to estimate the bounds on the load torque.

4.3.3 Eigenstructure Assignment Approach

The paper by Jørgensen *et al.* (1995) describes the design of fault detection observers in connection with both faults. Eigenstructure assignment is applied for the current fault detector.

A first order observer for detection of the position fault is designed similarly to the base-line observer (eq. 4.4), except that a fixed threshold is used. Different thresholds are selected for the different data-sets. The observer pole is chosen to L=-22.3 rad/s about twice as fast as the base-line design so the residual plots are roughly similar to those in figure 4.6, but with slightly faster responses. The detection delays for the different data-sets are listed in table 4.6.

An observer for detection of current faults is designed, where the right Eigenstructure assignment technique is used to decouple the position fault from the residual. The

Table 4.6: *Fault detection delays for the position fault detection algorithm designed by Jørgensen* et al. *(1995).*

Detection delay	DS1	DS2	DS3	DS4
Value of fixed threshold [rad]	0.002	0.002	0.002	0.046
Position fault, observer [msec]	40	40	40	120

unknown load input is entirely neglected in the design.

A standard discrete-time observer is used:

$$\hat{\mathbf{x}}(k+1) = \mathbf{A}\hat{\mathbf{x}}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{K}(\mathbf{y}(k) - \mathbf{C}\hat{\mathbf{x}}(k)), \tag{4.12}$$

where $\mathbf{A}, \mathbf{B}$, and $\mathbf{C}$ are the system matrices and $\mathbf{K}$ is the observer gain matrix. A residual is generated as a filtered version of the output estimation error:

$$r_a(k) = \mathbf{H}(q)(\mathbf{y}(k) - \mathbf{C}\hat{\mathbf{x}}(k)). \tag{4.13}$$

The basic idea with right Eigenstructure assignment is to design $\mathbf{K}$, so $\mathbf{K}\mathbf{F}_s$ becomes a right eigenvector of the observer feedback matrix, $\mathbf{G} = \mathbf{A} - \mathbf{K}\mathbf{C}$, where $\mathbf{F}_s$ is the entry matrix for the position fault, f_s. It is shown in the paper that this, together with a proper design of the output filter $\mathbf{H}(q)$ decouples the residual r_a from f_s. The resulting residual generator is of 4th order and not presented in the paper. A fixed threshold is applied, which leads to the detection delays listed in table 4.7. The current fault is not detectable

Table 4.7: *Fault detection delays for the current fault detection algorithm designed by Jørgensen* et al. *(1995).*

Detection delay [msec]	DS1	DS2	DS3	DS4
Value of fixed threshold [rad/sec]	8	11	12	242
Current fault, observer	120	130	130	∞

in DS4 because the threshold has to set high enough to avoid false alarms. Plots of the residuals with thresholds are shown in figure 4.10.

The paper also presents a method for the determination of the values of the fixed thresholds which can also be used for a subsequent analysis of robustness. The method is based on an experimental (or simulated) analysis of the standard deviation of the residuals when the system is subject to 1) normal set-point changes (n_{ref}), 2) changes in unknown input (Q_l), or 3) active fault events (Δi_m or Δs_o). The threshold for a residual, that is sensitive to a specific fault, is then determined as a trade-off between the standard deviation identified with the considered fault and the standard deviations of all the other inputs. The robustness analysis is performed by computing the ratio between the different standard deviations. Sufficient robustness is achieved if the standard deviation of the fault input is much larger than the standard deviations of the other inputs. It

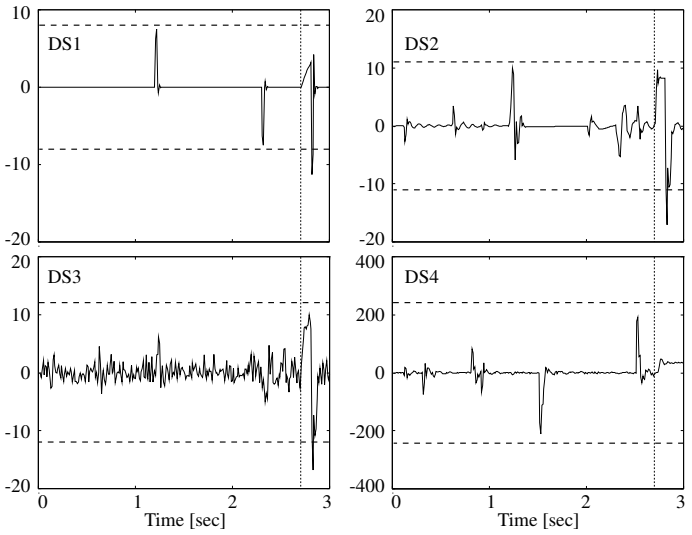


Figure 4.10: *Residuals for the current fault detection algorithm designed by Jørgensen* et al. *(1995).*

is found, by this method, that the position measurement faults are detectable, although perfect decoupling from the current fault is not achieved. It is also concluded that current faults are detectable, but the load input is a source of false alarms. It can, furthermore, be seen from the results in the paper that the standard deviation measures depend on the input signal selection. This is clear from data-set DS4, where the standard deviations are systematically different from the other data-sets, because this data-set is generated with larger velocity reference changes than the other data-sets.

It is recognized in the paper that the thresholds must be adaptive to cope with different excitation levels. This is further developed in the Ph.D thesis, Jørgensen (1995), where the standard deviation approach is developed into a technique for adaptive threshold design.

4.3.4 Parametric Statistical Approach

A statistical approach to the detection of both faults is presented in Grainger *et al.* (1995), where the change detection problem is solved with a set of sequential probability tests of the innovations from a bank of Kalman filters.

The occurrence of a fault is formulated as a change detection problem, where the nominal system parameters (θ_0) changes to the faulty system parameters (θ_1) at an unknown time step j. This is a problem of testing a hypothesis $\mathcal{H}_j$ against a hypothesis

$\mathcal{H}_0$, where

$$\begin{aligned} \mathcal{H}_0 &: \theta = \theta_0, \quad 0 \leq t \leq k \\ \mathcal{H}_j &: \theta = \begin{cases} \theta_0, & 0 \leq t < j \\ \theta_1, & j \leq t \leq k \end{cases} \end{aligned} \tag{4.14}$$

For each hypothesis, $\mathcal{H}_j$, a sequential probability test (SPRT) compares the likelihood ratio

$$S_k^j = ln \frac{p_j(y_0, ..., y_k)}{p_0(y_0, ..., y_k)}, \tag{4.15}$$

to a threshold h. The function p_j is the probability density of the innovations under hypothesis $\mathcal{H}_j$.

The likelihood ratio S_k^j is computed from innovations with associated variance estimates that are generated by a bank of Kalman filters. A recursive algorithm for this is provided in the paper. The decision function is given as

$$g_k = \max_{0 \leq j \leq k} S_k^j, \tag{4.16}$$

where a fault is declared active if g_k exceeds a threshold h. Explicit expressions for the alarm time and estimated fault onset time are provided in the paper.

The number of Kalman filters needed for calculation of the decision function is $k + 1$. The paper introduces a "pruning" rule that restricts the number of filters to a fixed number that for the benchmark example was chosen to 5. In the position measurement fault case, only one filter is required because the fault enters as an output fault. This is not the case for the current fault, so a total of six filters is used for the FDI.

The Kalman filters need an estimate of the unmeasured disturbance (Q_l). The paper proposes a generalized likelihood ratio (GLR) test to obtain an estimate $\hat{Q}_l$. It is assumed that the disturbance is a step function that changes from zero to Q_l at some time step j. The GLR test is performed in a fixed window of past samples. The step magnitude ($\hat{Q}_l$) and time of occurrence can be determined with a decision function $g_{k,d}$ that is compared to a fixed threshold h_d.

Linear Kalman filters are used for both position and current fault detection for all data-sets, except for the current fault detection of DS4. This data-set is generated under large input excitations and it has been necessary to apply a 6th order nonlinear observer, that includes the effect of current saturation and load dynamics, instead of a Kalman filter. A decision function is computed based on the observer estimation error. The unknown load input is not included in the nonlinear observer, so the above GLR test for load estimation is not performed in this case. The residuals are thus sensitive to load changes. As a solution for this problem, it is suggested to restart the testing when all residuals become large. An overview of the algorithms used for FDI is given in table 4.8.

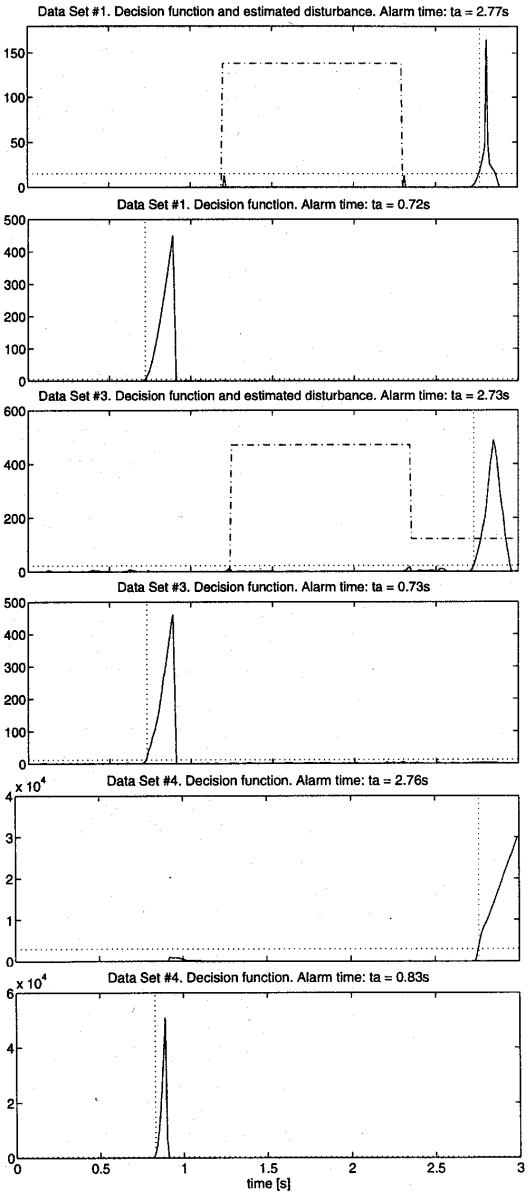


Figure 4.11: *Decision functions for current and position measurement faults for data-set DS1, DS3, and DS4, as presented by Grainger* et al. *(1995). Test threshold are horizontal dotted lines. Estimated load signal is dash-dotted line. The alarm times are indicated with vertical dotted lines.*

Table 4.8: *An overview of the algorithms used by Grainger* et al. *(1995) to detect position and current faults.*

Fault	Data-set	Algorithms
Position	DS1, DS3, DS4:	One 1st order linear Kalman filter; 5 SPRTs,
Current	DS1, DS3 :	Six 3rd order linear Kalman filters; 5 SPRTs; 1 GLR,
Current	DS4 :	Six 6th order nonlinear observers; 5 SPRTs; Test reset mechanism.

Table 4.9: *Fault detection delays for the position and current fault detection algorithms designed by Grainger* et al. *(1995).*

Detection delay [msec]	DS1	DS3	DS4
Position fault	20	30	90
Current fault	70	30	30

The performance of the FDI scheme is given by the detection delays listed in table 4.9. Plots of the decision functions g_k for the three data-sets are seen in figure 4.11

The design parameters are tuned differently for each data-set to provide optimal detection and avoid false alarms. The threshold for the large excitation case in DS4 is much higher than for the low excitation cases, and it is recognized that better performance can be achieved by improved models and faster sampling. The results show a very good decoupling between the unknown load input and the current fault.

4.3.5 Multiple Models Hypothesis Testing Approach

Earlier studies on detection of the current fault and isolation of load disturbances are presented in Bøgh (1995). The method is based on a hypothesis test of a model of the faulty system against a model of the non-faulty system.

Multiple models hypothesis test is used to test a number of observers, each representing a specific failure mode, against a 3rd order linear observer, representing the non-faulty system. This means that explicit information about influences of the faults on the system is utilized. The linear observer is designed with the Eigenstructure assignment approach (as described in section 4.3.3) with the purpose to increase sensitivity to the current fault and robustness to velocity measurement noise. The current fault is modelled as a "diode" (only positive current possible) and included in a 3rd order nonlinear continuous time observer together with the effects of current saturation and velocity controller integrator limits. The linear part of the nonlinear observer is designed similar to the linear observer using Eigenstructure assignment. The observer is executed using a fourth order Runge-Kutta algorithm with a fixed step length of 5 msec.

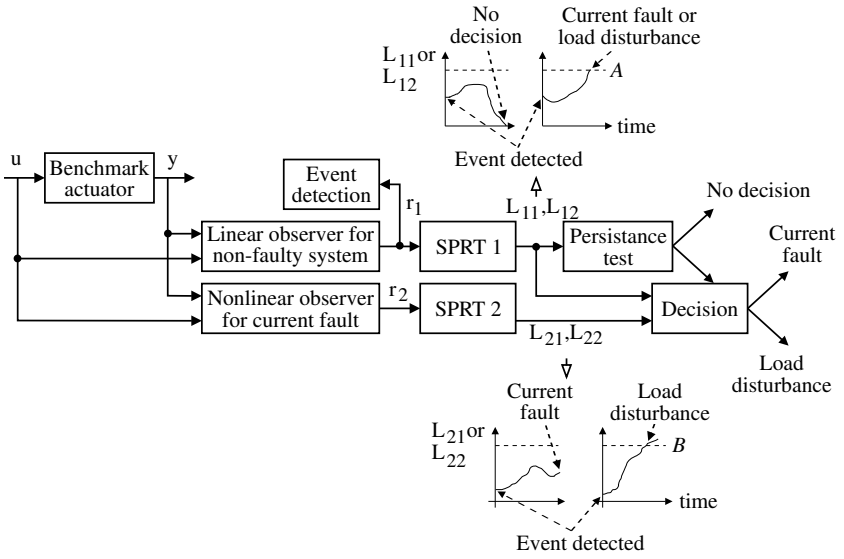


Figure 4.12: *Statistical test scheme for detection of the current fault and isolation to unknown load inputs. The decision is based on sequential probability ratio tests (SPRTs) of the residuals from two observers, one for the nominal non-faulty system, and one for the system with current fault.*

The residual from the linear observer is used to detect an event which is either an excitation of unmodelled dynamics, a current fault, or a load disturbance. When an event is detected, two SPRT algorithms begin processing the residuals from each observer with the purpose to isolate between the current fault and the load disturbance. This is illustrated in figure 4.12. The hypotheses for the two SPRTs are based on changes in the mean value of the residuals, μ_1 and μ_2, as given by table 4.10. Four log-likelihood variables are computed for this purpose to indicate either a positive or negative change in the average of the two residuals:

$$L_{j1} = \ln \frac{p(r_j^*|\mu_j = +\nu_j)}{p(r_j^*|\mu_j = 0)} \qquad L_{j2} = \ln \frac{p(r_j^*|\mu_j = -\nu_j)}{p(r_j^*|\mu_j = 0)}, \quad j = 1, 2 \tag{4.17}$$

where $p(r_j^*|\mu_j = m)$ is the probability density function for a window of past samples of the residual r_j under the hypothesis that the mean value is m. A recursive formula for the calculation of these decision variables is provided in the paper under the assumption that the residuals are Gaussian random sequences.

The outputs of SPRT1 (L_{11} and L_{12}) are used to determine if the effect of a detected event is persistent enough to draw a conclusion. If either L_{11} or L_{12} exceeds a threshold A then the event is persistent. If both L_{11} and L_{12} reaches zero, no decision is taken and

Table 4.10: *Hypothesis tests for the detection of current faults and isolation to unknown load disturbances for the two SPRT algorithms shown in figure 4.12.*

Test	Hypothesis	Rationale
SPRT 1	$\mathcal{H}_0 : \mu_1 = 0$	No event
	$\mathcal{H}_1 : \lvert\mu_1\rvert = \nu_1$	Event persistent
SPRT 2	$\mathcal{H}_0 : \mu_2 = 0$	Current fault active
	$\mathcal{H}_1 : \lvert\mu_2\rvert = \nu_2$	Load disturbance

the test is stopped. Otherwise, the test is continued until a conclusion is drawn. If the event was found persistent, the output of SPRT2 is then examined. If both L_{21} and L_{22} are below a threshold B, it is concluded that a current fault caused the event. If L_{21} or L_{22} exceeds B, a load disturbance is concluded.

This FDI scheme has been tested with different data-sets. The design parameters for residual mean and average were selected differently for the individual data-sets, which indicate a need to adapt to the signal excitation level. The resulting detection delays are shown in table 4.11. Results for DS5 are presented in figure 4.13. It it seen that the

Table 4.11: *Fault detection delays for the fault detection algorithm designed by Bøgh (1995).*

Detection delay [msec]	DS3	DS4	DS5
Current fault	50	50	40

current fault can be successfully detected without having false alarms caused by load changes. It is anyway, as anticipated by design, not always possible to detect the current fault when load changes appear simultaneously with the presence of current faults.

4.3.6 Parameter Estimation Approach with Extended Kalman Filter

As the only paper considering parameter estimation for FDI, Walker and Huang (1995) presents the application of extended Kalman filters for both position measurement and current fault detection.

The extended Kalman filter (EKF) has the form

$$\begin{bmatrix}\dot{\mathbf{x}}(t)\\ \dot{\theta}(t)\end{bmatrix} = \begin{bmatrix}\mathbf{f}(\mathbf{x}(t),\mathbf{u}(t),\theta,t)\\ \mathbf{F}_\theta\theta\end{bmatrix} + \begin{bmatrix}\mathbf{w}_1(t)\\ \mathbf{w}_2(t)\end{bmatrix} \tag{4.18}$$
$$\mathbf{y}(t_k) = [\mathbf{C} \vdots \mathbf{0}]\begin{bmatrix}\mathbf{x}(t_k)\\ \theta(t_k)\end{bmatrix} + \mathbf{v}(t_k),$$

where $\mathbf{x}$ is the system state vector with covariance $\mathbf{P}_x$, $\mathbf{f}$ contains a linear or nonlinear

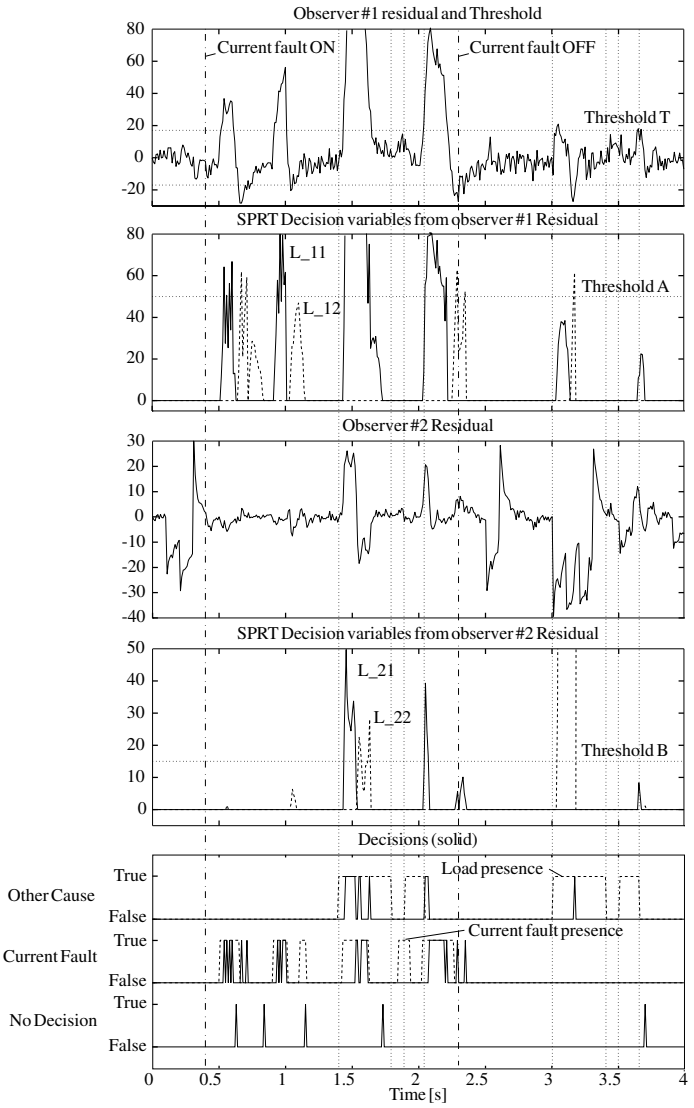


Figure 4.13: *Test results of the current fault detection scheme proposed in Bøgh (1995) for data-set DS5. Above are the residuals for the two observers: The linear observer for the non-faulty system r_1 and the nonlinear observer for the current fault r_2. These residuals are used to generate the decision variables L_{11}, L_{12}, L_{21}, and L_{22} that underlie the decisions shown in the bottom graph. The decisions are shown together with the periods of load and current fault presence. The current fault is only present when the desired current is negative.*

description of the system, θ holds the parameters to be estimated, $\mathbf{w}_1$ is process noise, $\mathbf{w}_2$ is a "pseudo-noise" that improves parameter convergence, $\mathbf{F}_\theta$ is a decay term for the parameter estimates that mitigates the problems with biased parameter estimates, $\mathbf{C}$ is the output matrix, and $\mathbf{v}$ is measurements noise.

The paper presents two approaches for comparison, a linear and a nonlinear EKF denoted EKF-L and EKF-NL respectively. EKF-L is based on the linear model (eqs.4.1-4.2) and has the state vector $\mathbf{x} = [i_2 \ n_m \ s_o \ x_{pos}]^T$, where the additional state x_{pos} is the internal state in the PI position controller. The parameters to be estimated are the position sensor bias f_s and the actuator current bias f_a, which are both supposed to be zero when the faults are not present. The parameter vector is then $\theta = [f_a f_s]^T$. The load input is considered unknown. The design of the nonlinear EKF-NL is based on a nonlinear model of the system, which includes the dominant nonlinearities and the dynamics of the load torque. This model is a simplified version of the full scale nonlinear model mentioned in section 4.2 and it is available in Bøgh *et al.* (1993)). The state vector is again augmented with x_{pos}, which makes the filter of 7th order. The parameter vector is extended to $\theta = [f_a \ f_s \ Q_{fl} \ I_l^{-1} \ \tilde{Q}_s \ Q_{fm}]$, where the additional parameters are, respectively, the friction torque (Q_{fl}) and inertia (I_l) of the load subsystem, the load torque ($\tilde{Q}_s$ which is the same as Q_l for the linear model), and the motor subsystem friction torque (Q_{fm}). Both Kalman filters are implemented with a standard algorithm. The integration of the state equations of EKF-NL is done with a 2nd order Runge-Kutta algorithm.

The EKF-approach includes a large number of parameters ($\mathbf{w}_1, \mathbf{w}_2, \mathbf{v}$, and $\mathbf{F}_\theta$) that are determined by inspection of the data-sets and from experience. Different values for the parameters are used for the different data-sets. The thresholds for the fault signatures (f_a and f_s) are computed on-line from $\pm C\sigma$, where σ^2 is the appropriate diagonal element of the state covariance matrix $\mathbf{P}_x$ and C is chosen by design. C is assigned different values for EKF-L and EKF-NL and also for the different data-sets.

The two Kalman filters are applied to data-set DS3 and DS4, where the threshold is tuned for each simulation. The tests on DS4 show poor performance with several false alarms so an additional test with fixed threshold is examined. Table 4.12 shows the results. The corresponding time histories of the current bias f_a and the sensor bias f_s are shown in figure 4.14. It can be seen that the decision signals are very sensitive to changes in the reference and load inputs and also subject to significant bias build-up. There is, furthermore, a poor isolation between the two faults.

The complexity of the EKF-NL algorithm is indicated in the paper. It requires several hours to simulate the 3 seconds data-set on a time-shared IBM RS6000 workstation.

4.3.7 Interacting Multiple Model Approach

An alternative approach using Kalman filters based on statistical decision theory is presented in Efe and Atherton (1997). The paper presents the interacting multiple model (IMM) technique and shows it's ability to detect and isolate both faults in some special

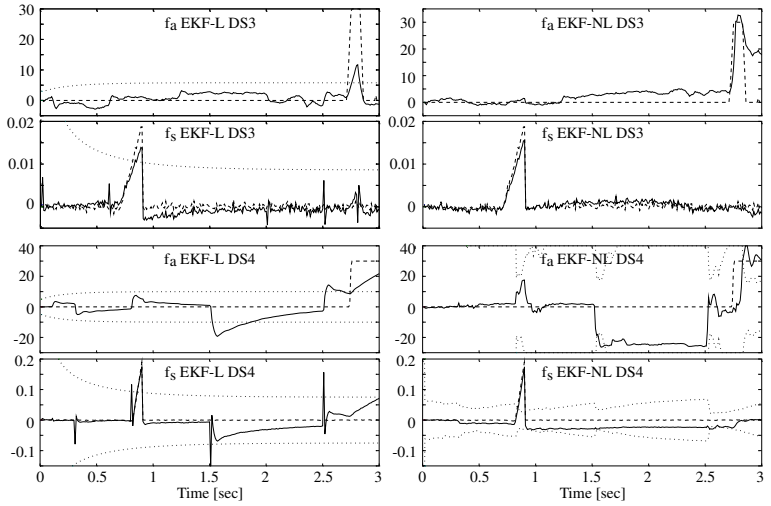


Figure 4.14: *Fault detection signals for the FDI scheme proposed by Walker and Huang (1995) applied to DS3 and DS4. The graphs show the current bias f_a and the sensor bias f_s for a linear Kalman filter (EKF-L) and a nonlinear Kalman filter (EKF-NL). Dotted lines are decay terms, solid lines are estimated biases, and dashed lines are actual biases.*

Table 4.12: *Fault detection delays for the position and current fault detection algorithms designed by Walker and Huang (1995). The table shows the detection delays and false alarm rates for the different design approaches and also the different choices of the decision threshold.*

Data-set →	DS3			DS4		
Threshold - Det. delay - False alarms →		[msec]	[%]		[msec]	[%]
Position fault, EKF-L	1.8σ	40	-	5σ	70	1
Position fault, EKF-NL	5σ	40	-	15σ	110	1.7
Position fault, EKF-NL, fixed threshold	-	-	-	0.03 rad	110	0.3
Current fault, EKF-L	3σ	60	-	5σ	40	17
Current fault, EKF-NL	5σ	40	-	8σ	110	2
Current fault, EKF-NL, fixed threshold	-	-	-	26 A	100	1

cases.

The IMM algorithm uses a bank of Kalman filters where one filter represents the nominal model and the remaining filters represent dedicated fault models. Probabilities of each model being correct are calculated from the filter innovations and associated covariances. These probabilities are used to mix the internal states between all the Kalman filters before the state update of the filters are performed. The IMM algorithm works as

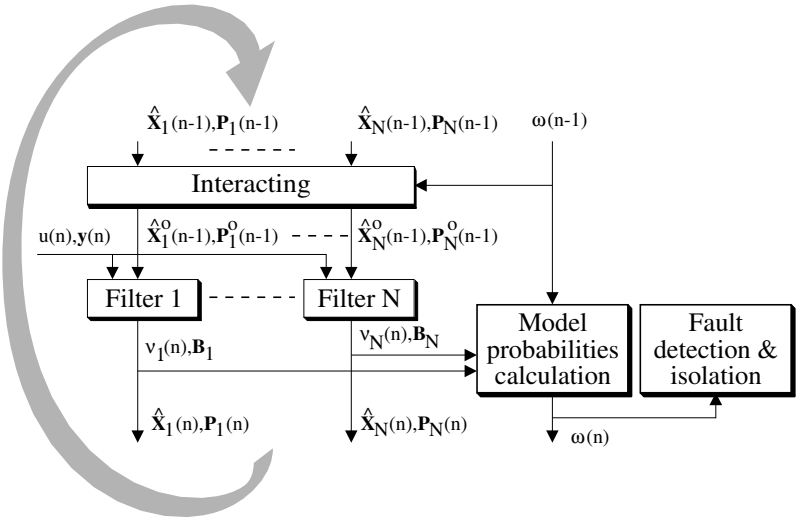


Figure 4.15: *The interacting multiple model (IMM) algorithm used for FDI (reproduced from Efe and Atherton (1997)).*

depicted in figure 4.15. The figure shows the actions performed before and after update of the N Kalman filters (from time-step $n-1$ to n). The Kalman filter states and state covariance matrices ($\hat{\mathbf{X}}_j(n-1)$ and $\mathbf{P}_j(n-1)$) are mixed (hence the name "interacting") in proportion to the individual model probabilities $\omega(n-1)$ by

$$\hat{\mathbf{X}}_j^o(n-1) = \sum_{i=1}^{N} \frac{\mathcal{P}_{ij}\omega_i(n-1)}{\sum_{l=1}^{N}\mathcal{P}_{lj}\omega_l(n-1)}\hat{\mathbf{X}}_i(n-1) \tag{4.19}$$

$$P_j^o(n-1) = \sum_{i=1}^{N}\Bigg\{\frac{\mathcal{P}_{ij}\omega_i(n-1)}{\sum_{l=1}^{N}\mathcal{P}_{lj}\omega_l(n-1)}\Big[P_i(n-1) + \tag{4.20}$$

$$(\hat{\mathbf{X}}_i(n-1) - \hat{\mathbf{X}}_i^o(n-1))(\hat{\mathbf{X}}_i(n-1) - \hat{\mathbf{X}}_i^o(n-1))^T\Big]\Bigg\},$$

where $\mathcal{P}_{ij}$ is a transition probability matrix with dominant values along the diagonal. It is selected in advance and remains constant. The filter innovations are denoted ν_j and the associated covariance matrices denoted $\mathbf{B}_j$. Fault detection is performed on the model probability time histories. If the probability from a particular fault model exceeds a fixed threshold (set to 0.15 for the benchmark), the corresponding fault is declared. Further details on the IMM algorithm is found in the paper.

The current fault detection problem is solved using three Kalman filters, each with a different amplitude of the current fault vector f_a:

Model #	1	2	3
Current fault f_a	0	30 A	40 A

Model 1 represents the nominal system whereas model 2 and 3 represent different amplitudes on a positive current fault. Model 2 is applied for DS1 and model 3 for DS2-DS4. Model 2 successfully detects the current fault in DS1, but if it is applied to DS3, false alarms will be present (these graphs are not shown here). Examples on the time histories of the probability of model 3 applied to DS3 and DS4 are shown in figure 4.16. Graph

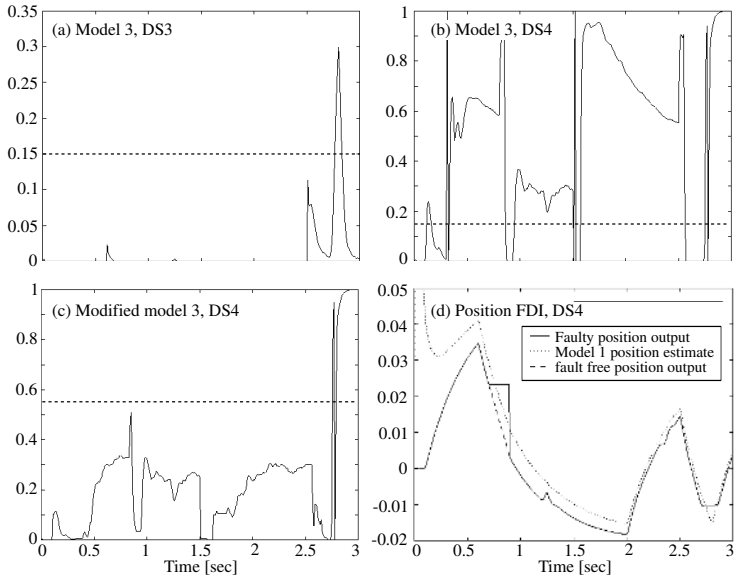


Figure 4.16: *Probability time histories for the interacting multiple model FDI approach presented by Efe and Atherton (1997). Current fault detection signals are shown in (a), (b), and (c), whereas (d) shows the signals for position fault detection. Vertical dashed lines are decision thresholds.*

(a) shows successful detection of the current fault but also sensitivity to negative velocity reference changes (e.g. spike at 2.5 sec). By inspection of the graphs in the paper, the reason for this is found to originate from negative current spikes that match the modelling of the fault (f_a = 40 A). It is also clear that load changes are able to cause those changes in the model 3 probability. This conclusion is not addressed by the authors. The sensitivity problem is even worse for the large excitation case of DS4 in (b), which is clearly inadequate for detection. To solve this problem, the authors suggest a modification to the algorithm, so the model probability update is suppressed if the probability

exceeds the threshold simultaneously with a velocity reference change greater then 333 rad/s^2. The detection scheme for the corresponding sample is then skipped. The resulting model 3 probability time history is seen in (c). This shows improved performance relative to (b), but also a need for a larger threshold.

Position fault detection requires no additional fault models, but is achieved by comparing the position estimate of model 1 to the position measurement $s'_{o\nu}$. The estimated position, plotted in figure 4.16 (d), is seen to converge slowly, and will possibly be applicable for fault detection over time. The performance is not clear from the paper. It can be seen that this position FDI method is sensitive to current faults so isolation between the two faults is not achieved. The thresholds selected for position fault detection are different for low and high excitation: 0.005 rad for DS1-DS3 and 0.09 rad for DS4.

The resulting detection delays, presented in table 4.13, are claimed in the paper although they are not all verifiable from the presented material.

Table 4.13: *Fault detection delays for the position and current fault detection algorithms designed by Efe and Atherton (1997).*

Detection delay [msec]	DS1	DS2	DS3	DS4
Position fault	20	40	50	100
Current fault	30	20	60	20

4.3.8 Parity Equations Approach

The parity space method is considered in Mediavilla *et al.* (1997) for the detection of the current fault. As time-invariant linear methods are incapable to detect the current fault, the paper suggests the parity space approach for the design of a *time-variant* linear filter that decouples the multiplicative current fault from the additive load disturbance. The parity space approach is also applied in section 4.3.1 for position fault detection.

The design of parity equations is considered both by input-output models and by state-space models. The parity equation for the state-space approach is similar to eq. 4.5 except that the transformation vector $\mathbf{v}$ is made time-variant. $\mathbf{v}(t)$ is designed to decouple the residual $r(t)$ from states and load torque by

$$\mathbf{v}(t)^T \mathbf{T} = 0, \tag{4.21}$$

$$\mathbf{v}(t)^T \mathbf{Q}_E = 0. \tag{4.22}$$

The residual shall, furthermore, be sensitive to the multiplicative fault. The multiplicative fault leads to changes in the system matrices, $\Delta\mathbf{A}$, $\Delta\mathbf{B}$, and $\Delta\mathbf{C}$, which again lead to changes in $\mathbf{T}$ and $\mathbf{Q}_E$ of $\Delta\mathbf{T}$ and $\Delta\mathbf{Q}_E$. It is shown in the paper, that optimal sensitivity to the multiplicative fault is obtained by maximizing

$$\mathbf{v}(t)^T \mathbf{s}(t), \tag{4.23}$$

where

$$\mathbf{s}(t) = \Delta\mathbf{T}\mathbf{x}(t-k) - \Delta\mathbf{Q}_E \begin{pmatrix} u(t-k) \\ \vdots \\ u(t) \end{pmatrix}. \tag{4.24}$$

This means that $\mathbf{v}(t)$ can be chosen as one of the vectors, spanned by the subspace described by eq. 4.22, that is most parallel to $\mathbf{s}(t)$. This problem is solved by using least squares minimization. An additional problem is that $\mathbf{s}(t)$ depends on the state variables (i_2 and n_m) which cannot be measured nor estimated without being effected by the faults. A solution to this problem is suggested where $\mathbf{s}(t)$ is estimated from data-set DS1. $\mathbf{s}(t)$ can easily be calculated using the linear model applied for DS1, so an average value of $\mathbf{s}(t)$ during the current fault period 2.7-3.0 sec is calculated and used for the other data-sets. An observation, which is not stated in the paper, is that the constant value of $\mathbf{s}(t)$ leads to a time-invariant detection filter which makes it theoretically impossible to decouple the load disturbance.

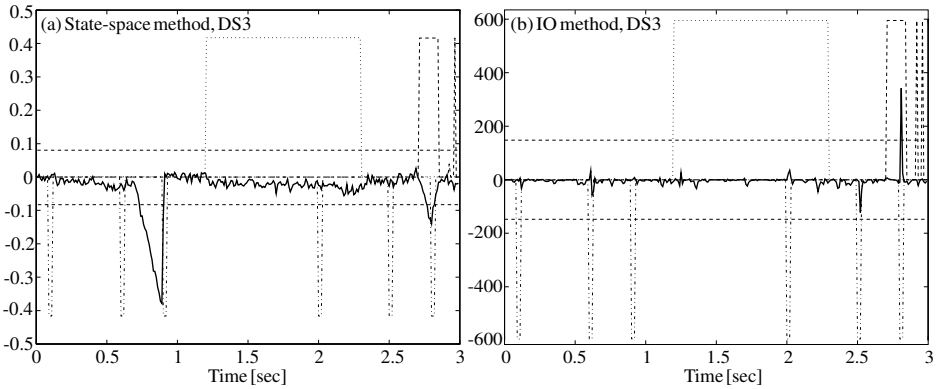


Figure 4.17: *Residuals for the current fault detection algorithm proposed by Mediavilla* et al. *(1997). Graph (a) shows the cumulative sum of the primary parity space residuals from the state-space approach (solid). Graph (b) shows the primary residuals from the input-output approach (solid). Indicators are shown for periods with load disturbance (dotted), active current faults (dashed), and significant changes in reference (dash-dotted). Horizontal dashed lines are thresholds..*

The state-space approach is tested on DS1-DS4, where the cumulative sum of the residuals are used as detection signals. The result for DS3 is shown in figure 4.17 (a), where it is clear that the current fault can be detected but not isolated from the position fault. The residuals show some sensitivity to load torque inputs but almost no influence from velocity reference changes.

The input-output model of the system with an additive load input ($d = Q_l$) and a multiplicative fault is described by

$$(\mathbf{H}(q) + \Delta\mathbf{H}(q))\mathbf{y}(t) = (\mathbf{G}(q) + \Delta\mathbf{G}(q))u(t) + (\mathbf{G}^s(q) + \Delta\mathbf{G}^s(q))d(t), \quad (4.25)$$

where $\mathbf{G}$ and $\mathbf{H}$ are the nominal transfer functions and $\Delta\mathbf{G}$ and $\Delta\mathbf{H}$ represent the changes in $\mathbf{G}$ and $\mathbf{H}$ caused by the current fault. A residual is generated as a filtered version of the primary parity residuals, which gives the analytical equation

$$\begin{aligned} r(t) &= \mathbf{W}(t)\mathbf{o}(t) = \\ &\mathbf{W}(t)\Big(\Delta\mathbf{G}(q)u(t) - \Delta\mathbf{H}(q)\mathbf{y}(t) + (\mathbf{G}^s(q) + \Delta\mathbf{G}^s(q))d(t)\Big). \end{aligned} \quad (4.26)$$

The transformation $\mathbf{W}(t)$ is designed to obtain the desired properties. Decoupling from additive inputs leads to a constant polynomial $\mathbf{W}(q)$ whereas decoupling from multiplicative inputs (as in this case) leads to a time-varying matrix (see also the introduction to parity space on page 14). By experiments, the authors recognized lack of freedom in the design of $\mathbf{W}(t)$ to isolate the additive load input from the multiplicative fault. Instead, the residuals are maximized with respect to the fault. Results from simulations for all data-sets are available in the paper and as an example, DS3 is seen in figure 4.17 (b). It can be seen from the results that the residual is sensitive to both load disturbances and velocity reference changes, but in contradiction to the state-space residuals, insensitive to position faults.

Both the state-space and the input-output approaches are able to detect the two faults for data-sets DS1-DS3, but the input-output method is not able to isolate between the two faults. Both methods are incapable to detect faults for the high excitation signals in DS4 because they are based on linear design. Current saturation effects cause several false alarms for this case. The final results are given as the detection delays presented in table 4.14.

Table 4.14: *Fault detection delays for the current fault detection algorithm designed by Mediavilla* et al. *(1997).*

Detection delay [msec]	DS1	DS2	DS3	DS4
State-space residuals	140	40	40	∞
Input-output residuals	130	110	110	∞

4.3.9 Approach using Neural Networks

An alternative to the model-based techniques is presented by Köppen-Seliger and Frank (1996) that applies a neural network (NN) to detect and isolate the two faults, even when they appear simultaneously.

The NN technique has previously been applied to FDI problems in slowly varying processes, but Köppen-Seliger and Frank (1996) illustrates feasibility also for fast processes such as the benchmark equipment. Two approaches are presented. One is a scheme of two NNs, where the first NN is trained to imitate the behaviour of the process for residual generation, and the second NN is trained to evaluate the residuals for classification of the faults. The other approach is a one-step diagnosis (OSD), where a single NN is trained to classify the faults directly from measured signals.

The NN for residual generation is based on a radial-basis-function (RBF) network trained with the gradient-descent method to reproduce the nonlinear mapping function $\mathbf{g}$ of the system's input-output form:

$$\mathbf{y}(k) = \mathbf{g}(\mathbf{y}(k-1), ..., \mathbf{y}(k-q), u(k), ..., u(k-p)). \tag{4.27}$$

The training data represents the nominal system without faults. The network is then used to generate residuals and the result is presented in the paper for a position fault case. The residual shows sensitivity to the fault but also a drift in the non-faulty period that is large enough to cause false alarm. This is not further explored in the paper and the second NN for residual evaluation is therefore not considered.

The OSD design is based on a restricted-coulomb-energy (RCE) network that has the ability to add neurons depending on the complexity of the underlying problem. The NN is trained with a binary output pattern reflecting the fault categories of the input vector. The input vector consists of the velocity reference n_{ref}, the measured position $s'_{o\nu}$, and the motor velocity $n_{m\nu}$. The training is performed with different combinations of reference input, load disturbances, and fault scenarios. Results are presented for two fault scenarios, one where the position fault and the current fault occur in different time periods, and one where the two fault periods overlap. In both cases, the network is able to detect and isolate correctly. The data-sets are different to those suggested in the benchmark proposal but the excitation level is comparable to the low amplitude cases DS1-DS3. No results for high excitation as in DS4 are provided.

The paper recognizes the inevitable problem of black box modelling, that the training input must represent literally all possible operating conditions including fault situations. In practice, fault scenarios are not available from the real process, so the OSD approach necessitates the development of a sufficient accurate simulation model. The NN in the OSD scheme has, furthermore, increased complexity and training time compared to the residual evaluation network because the raw input signals vary continuously within the entire operational range whereas the residuals are nominally close to zero.

4.4 Discussion

The nine contributions to detection and isolation of the position measurement and current faults in the benchmark actuator, described in the previous section, represent a wide range of approaches. This gives an excellent opportunity to directly compare the fea-

sibility of different FDI methods applied to typical failures in fast electro-mechanical equipment.

4.4.1 Residual Generation

The main categorization of the residual generation techniques is into methods based on *parameter estimators*, *state observers*, and *parity space equations*. Each approach has it's pros and cons although similarities exist.

The parameter estimation methods are inadequate for the present problem, because the two considered faults have a very abrupt effect on the system variables. The current fault, modelled as an additive input, has almost the characteristics of a step function and the position fault act as a ramp with a relatively steep slope. Generally, parameter estimation techniques are slow and more appropriate for detection of incipient faults. This is also pointed out in Höfling *et al.* (1995). The difficulties with parameter estimation are illustrated by the extended Kalman filter approach presented in section 4.3.6. The estimated position and fault parameters are continuously subject to undesirable biases, even after means have been taken to avoid this. These biases cause false alarms although high thresholds are chosen. The high thresholds lead to large detection delays. Furthermore, some of the detection delay values stated in the paper can be seen from the graphic material to be a matter of coincidences: First, a current fault is detected very quickly because the detection signal shows a favourable bias immediately before the fault becomes active. Second, a position fault is detected in the same sample where it is manifested in the signal, caused by what is most reasonable called a false alarm. In spite of these problems, the position detection signal seems appropriate for detection without false alarm even though the presented results operate with false alarms. A higher threshold will solve this issue but with a penalty on the detection delay.

An advantage with the parameter estimation approaches is that the fault amplitude is automatically estimated, if this is desirable. It is not entirely the case for the estimates in section 4.3.6 because the bias compensation (decay term F_θ) tends to reduce the estimated size.

The parity equations approach treated in section 4.3.8 (and also briefly considered in section 4.3.1) provides a nice and systematic approach for decoupling between faults and exogenous inputs. A drawback with the basic method is that the residuals suffer from a shortage of low frequency information. Detection of faults from the residual in figure 4.17 (b) is based on spikes in a single sample, which is inadequate for a proper decision. A similar residual is given in Höfling *et al.* (1995) for the position fault, that shows a small bias during the ramp period and a large spike when the fault disappears with a step. A solution to the problem is to process the primary residuals with a low pass filter. This makes the parity space design equivalent to the observer design as pointed out in the introduction to FDI on page 15. It is also possible to improve the detection by applying some statistical testing technique. The approach taken in Mediavilla *et al.*

(1997) for the state-space residuals is to use the cumulative sum of the residuals (see figure 4.17 (a)).

Observer based approaches are considered in Höfling *et al.* (1995) and Jørgensen *et al.* (1995) for a direct design similar to the base-line observer in eq. 4.4, in García *et al.* (1995) for an $\mathcal{H}_\infty$-design that separates the two faults, and in Jørgensen *et al.* (1995) for an Eigenstructure assignment of a current fault detector that is decoupled from the position fault. The various design approaches and parameter selections lead to different eigenvalues but also a different number of observer poles. The $\mathcal{H}_\infty$-design in García *et al.* (1995) lead to a 3rd order filter with three poles in -100 rad/sec for the position fault detector. Compared to the 1st order base-line observer, no significant improvement is gained from the higher order. The 3rd order filter is furthermore too fast to facilitate rapid detection of the position fault's ramp effect.

A common challenge for all the residual generation approaches is that the characteristics of the load disturbance is unknown. Additional knowledge about the load would have been helpful for isolation to the current fault. Some approaches therefore assume specific properties of this unknown input. The statistical approach presented by Grainger *et al.* (1995) assumes the load torque to be a step input which may not be the case. The IMM strategy presented by Efe and Atherton (1997) indirectly assumes positive load torque inputs. This detection scheme is sensitive to large positive additive current fault inputs (f_a), so large negative load disturbances may lead to false alarms. This situation was not represented in the proposed data-sets and therefore not considered in Efe and Atherton (1997).

4.4.2 Residual Evaluation

The task of residual evaluation is basically approached in two ways, either by threshold test or statistical testing. The latter is applied in both Grainger *et al.* (1995) and Bøgh (1995) as a sequential probability ratio test (SPRT) of observer residuals. An advantage with SPRT is that the thresholds can be designed directly from requirements to the ratio between false alarms to missed alarms, provided the statistical properties of the residuals are known. Normally, the residuals are assumed to be Gaussian random sequences, which is often an inadequate approximation to real life.

An important detail in the application of SPRT can be observed: The Kalman filters in Grainger *et al.* (1995) directly provide an estimate of the variance of the residuals (or innovations) that is used by the SPRT algorithm. The approach presented by Bøgh (1995) is based on observers that do not produce variances, so these are treated as fixed design parameters. This means that the latter approach is more sensitive to changes in the operational conditions.

It is a general experience that the dominant unmodelled nonlinear effect of current saturation together with a low sampling rate constitute the major sources for FDI perfor-

mance degradation. The typical solution is to improve the detection scheme by making the residual evaluation task dependent on changes in the position or velocity reference signals, which cause the nonlinear effects to develop. These problems are approached either with adaptive thresholds as suggested in Höfling *et al.* (1995) and García *et al.* (1995) or by altering the detection logic as proposed in Efe and Atherton (1997) and also included as an idea in Walker and Huang (1995). Also Grainger *et al.* (1995) uses a kind of online adaptation in the resetting mechanism that clears the detection variables if all residuals becomes large. It must be realized that these modifications lead to a varying sensitivity to faults and may cause missed alarms, especially if the reference input activity is high. Generally, it is preferable to shape the residuals with the additional information (if possible) and maintain a fixed threshold.

4.4.3 Standard Versus Dedicated Techniques

Some of the approaches apply a standard technique whereas others examine the structure of the system and design dedicated FDI schemes that suit the problems. The results show that the dedicated schemes often produce improved residuals compared to the standard techniques because specific details can be utilized and certain problems can be avoided. As an example, consider the position fault detection schemes that take all available signals as input (Walker and Huang (1995); Efe and Atherton (1997)). The residuals show sensitivity to current saturation and load effects, which is an unnecessary source to false alarms. These effects are easily decoupled by using only the velocity and position measurements in FDI, as suggested with the base-line observer in eq. 4.4. It is interesting to observe that the frequency domain approach presented in García *et al.* (1995) follows a strict recipe and still ends up with decoupled design, where the position FDI uses only the velocity and position measurements and the current FDI uses only the velocity reference and the velocity measurement.

4.4.4 Black-Box Neural Net Approach

The one-step diagnosis (OSD) approach using neural networks presented by Köppen-Seliger and Frank (1996) differs from the other methods, because residual generation and evaluation is combined into one algorithm. This makes it difficult to evaluate the performance with respect to fault sensitivity, isolability and robustness. The tests presented in the paper are performed with small reference input changes that do not trigger the dominant nonlinear characteristics of the actuator. As neural networks are known to be excellent to reproduce nonlinear mappings, it would have been interesting to investigate the performance for large input excitations.

A general problem with learning based methods for FDI is that long training sequences must be available that represent virtually all relevant combinations between operating conditions, external influences, and fault scenarios. It may be possible to collect these data in the laboratory setup of the benchmark actuator, but in many real-life processes it is not feasible to artificially introduce faults. Instead, mathematical models

are developed to generate simulated sequences that can be used for training and verification of the networks. The question is now, why a black-box model shall be trained to reproduce an existing and known mapping.

4.5 Conclusion

In general, the results show that the position measurement fault is detectable with linear methods and that no improvement is gained by applying nonlinear techniques. The current fault is detectable with linear methods, but can only be decoupled from the unknown load torque input with nonlinear or time-varying techniques, or if more information about the unknown input is assumed. Only the current fault detection technique presented in Bøgh (1995) successfully discriminates the current fault from the load input without assumptions on the unknown load torque.

Furthermore, most of the suggested FDI schemes have been designed with different parameters for the small signal data-sets (DS1-DS3) and the large excitation data-set (DS4). The common problem is that robustness to unmodelled dynamics stimulated by the large velocity changes in DS4 lead to unacceptable detection delays or missed alarms for the smaller fault effects in DS1-DS3. In practice, no á priori information can be assumed on the operation of the actuator (the ship may be in open sea or manoeuvring in harbour) so the FDI scheme shall operate in all situations. This means that the mentioned approaches must be further developed to adapt to different operational conditions.

An unambiguous comparison between the different approaches is impossible because different design decisions are taken. Some results show that the faults can be detected but not isolated. Others are able to catch the current fault but not distinguish to load disturbances. Different methods show different trade-offs between detectability of incipient faults versus false alarm rate. Most of the methods use different design parameters for the results obtained for the different data-sets, which means that the design is not mature. With these conflicts in mind, a summary of the detection delays, an assessment of the executional complexity, and the abilities to isolate are presented in figure 4.18. It is noticeable that the application of a more advanced algorithm does not necessarily lead to improved results. Instead, it seems advantageous to examine the structure of the system before the FDI design method is selected.

From the material involved in the benchmark test, it is concluded that the trivial position measurement fault is easily detected and isolated using the base-line observer presented in eq. 4.4 combined with an adaptive threshold test as suggested in Höfling *et al.* (1995). The more troublesome current fault is best detected with the application of nonlinear observers, where the approach described in Bøgh (1995) is superior taking run-time complexity of the algorithm into consideration. Although this method has not been designed to isolate the position fault, it proves it's potential in producing satisfactory short detection delays with a rather simple algorithm. The algorithm is able to distinguish load torque disturbances from current faults, which has only been achieved by one other approach, namely the fairly complex statistical approach by Grainger *et al.* (1995).

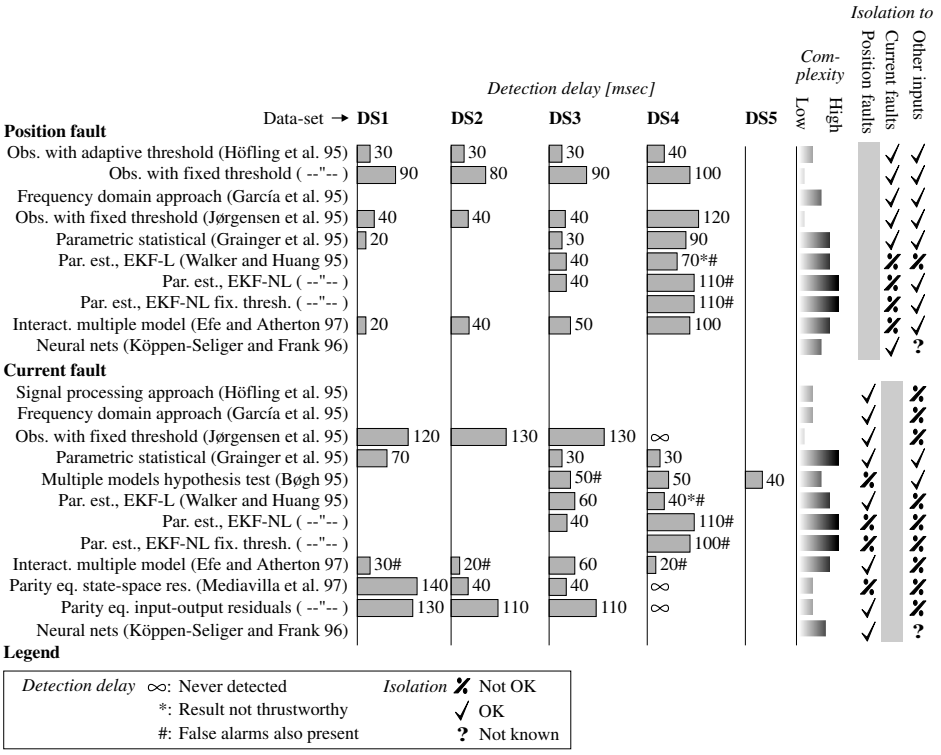


Figure 4.18: *Comparison of the performance of the fault detection algorithms presented in this chapter. The figure shows the detection delays for each data-set, the executional complexity of the algorithms, and the abilities to isolate the two faults and avoid false alarms from other exogenous inputs as load disturbances and large reference changes.*

4.6 Summary

The applicability of several methods for FDI were compared in this chapter based on studies using the same sets of data sequences from the benchmark equipment. In general, the investigation of the results showed that realistic problems like low sampling rate, saturation effects, and different operating conditions play an important role in the design. The standard techniques available today do not systematically account for these effects, so different ad-hoc solutions were suggested. Most FDI methods today are based on linear models, but to handle the real-life problems, nonlinear techniques must be applied. It was also shown that it is more beneficial to examine the structure of the system before FDI design than just applying a standard method to all available inputs and outputs.

The benchmark application belongs to the category of systems that have fast dynamics and where faults appear abruptly. Some FDI methods showed up to be inadequate for the benchmark case, but this does not mean they are inappropriate for other applications. The basic benchmark model used for FDI design is of only 3rd order, and most FDI algorithms are developed to take advantage of models of higher order to decouple disturbances and isolate between faults. This means that the benchmark case does not illustrate the full potential of the FDI methods, but it is a challenging realistic example.

Chapter 5

Fault Tolerant Control for the Ørsted Satellite

The second application example of this thesis is the attitude control system (ACS) of the Ørsted micro satellite. This chapter presents the design and implementation of fault tolerance in the control system using the analysis and design methodology presented in chapter 3. Earlier results on the Ørsted FTCS design have been published in Bøgh *et al.* (1995). The satellite represents a more complex system than the benchmark application so it is well suited to emphasize features of FTCS design that have not been addressed so far in the thesis.

The main emphasis of this chapter is on the design of the fault handling decision logic and the implementation in the three layer supervisor structure. Examples are given on how to structure the decision logic's state-event machines and how the design can be verified. The implementation of a supervisor in multitasking software is discussed in connection with synchronization to a control level task. Finally, a presentation of the ACS test procedures are used to illustrate how FTCS aspects can be tested and how an appropriate FTCS design supports testability.

5.1 Motivation and Related Work

There is a clear motivation for considering fault tolerance in ACSs for satellites. Wimmer (1997) reviews four ESA mission cases flown in the 1980s and 1990s, where each mission experienced roughly 150 anomalies. The attitude and orbit control system were responsible for 25-50% of all anomalies, several being mission critical. Many anomalies were related to software in the ACS and command and data handling (CDH) subsystems, because they have been made unnecessarily complicated. The reason for this was that digital avionics has been recently introduced in space, so experience and systematic

methods were lacking. Among the lessons learned, Wimmer stresses the need to design for simplicity, include in-orbit flexibility for introducing work-around solutions, and allow adjustments of the control- and fault management processes. He emphasises the necessity to include fault tolerance design early in connection with operational aspects.

The major developments of fault tolerance in space have been concentrated on fail-safe aspects in connection with large and expensive missions. The analysis techniques applied are very involved and require extensive manpower. The ESA standards for space product assurance of dependability (ESA-ECSS (1996*a*)) and safety (ESA-ECSS (1996*b*)) illustrate the complexity. These documents instruct the contractor to perform tasks such as function criticality assessment; failure mode, effects, and criticality analysis (FMECA); reliability modelling; availability analysis; hazard analysis; and probabilistic safety risk assessment. Fulfillment of these standards require expertise and experience, and will lead to a vast amount of documentation. This procedure is not applicable for small short-term projects as Ørsted.

To fulfill the fail-safe requirements, large satellites employ hardware redundancy and highly distributed architectures. An example is the $112 million 805 kg NEAR asteroid rendezvous spacecraft launched in February 1996 (Santo *et al.* (1995); Lee and Santo (1996)). The attitude and guidance control system uses 2-redundant computers, 2-redundant interface units, and a collection of partially redundant sensor and actuators that are dedicated for control only. Such architectures provide means for advanced cross-check and health monitoring strategies that are not possible in small satellites like Ørsted. The Ørsted ACS uses the science instruments for attitude determination and the ACS software is integrated with the CDH software on a single computer. It is not feasible to let the CDH subsystem watch the ACS performance and prevent malicious events, as is done on the NEAR spacecraft. This means that the ACS must be inherently fault tolerant and designed resistant to mission critical failures.

Recent advances in fault tolerance for space applications are found in connection with interplanetary missions. These missions are characterised by long communication delays, periods without ground contact, and goal-oriented mission objectives. The goal-oriented aspect means that the spacecraft shall be turned and pointed in different directions during different mission phases. On-board autonomy is implemented that optimizes the scheduling of activities in connection with planned manoeuvres, unanticipated spacecraft anomalies, and changes in the environment. The Saturn-bound $3.3 billion 5.6 ton Cassini spacecraft launched in October 1997 (Brown *et al.* (1995); Slonski (1996)) represents state-of-the-art in autonomy for satellites that are flying today. On-board rule-based reasoning is used for fault handling combined with a bank of error monitors and hard-coded fault responses. The New Millennium Project (NMP) Deep Space 1 (DS-1) spacecraft (Pell *et al.* (1997)), that is scheduled for launch in mid 1998, exceeds the autonomy capabilities of Cassini by applying AI-techniques for on-board constrained-based planning and scheduling together with model-based fault diagnosis.

Micro satellites like Ørsted are, in contrast to the above mentioned spacecraft, lightweight and cheap constructions. The Ørsted budget for development and operation is

$20 million (this amount actually exceeds typical micro satellite projects because it subsumes a relatively large amount of basic research). Expenses are kept low by avoiding the use of elaborate analysis methods like criticality assessment and reliability modelling and thus accepting reduced reliability assurance. It is not required that the satellites are fail-safe on the same level as expensive satellites, so the level of redundant hardware is kept low. Furthermore, the Ørsted mission is not goal-oriented as the planetary missions. The single task of the ACS is to stabilize the satellite throughout the entire mission. This means that the ACS is simpler and thus justifies the use of less complex development approaches and implementation techniques.

The approach for fault tolerance of the Ørsted ACS, presented in this chapter, is aimed at simplicity and flexibility. It is simple in the way, that the fault analysis is not based on formal use of reliability engineering. The method captures the necessary behaviour of the system during fault events in a simple description, that supports the implementation of fault tolerance, but without an associated estimate of the probability of system failure. The implementation is kept simple by organising independent functionalities into separate modules. The on-board decision taking is hard-coded, in contrast to rule-based reasoning, but designed with a systematic method, that ensures flexibility in both the design phase and the in-orbit maintenance phase. The use of systematic correctness verification and automated code generation ensures that the reliability of the FTCS is not compromised during design iterations and maintenance updates.

5.2 Introduction to the Ørsted Satellite

The Danish Ørsted satellite is scheduled for launch in spring 1998 from Vandenberg, California. The science mission is related to the geomagnetic field and its interaction with the solar wind plasma. It is launched into a 450 $\times$ 850 km orbit with a 96 degree inclination by a Delta II launcher. After release the ACS captures the satellite from a random tumbling and subsequently an 8 meter boom is deployed. In the remaining part of the mission the ACS stabilizes the satellite in three axes with the boom pointing away from the Earth. The performance requirements for ACS are to maintain the attitude within ± 10 degrees in pitch and roll (angular deviation of the boom from vertical), ± 20 degrees in yaw (rotation about the boom) and an angular velocity below 10 degrees/minute.

The configuration of the satellite is shown in figure 5.1. The science instruments are located on platforms displaced from the main body with the purpose to minimize EMC interference. Table 5.1 summarises the assignment of the on-board sensors. The ACS is an integrated part of the satellite where science instruments are used for attitude acquisition as far as possible instead of dedicated ACS instruments. This reduces the weight for the spacecraft platform and thereby improves the scientific potential. Two 16 MHz 80186 processors provide on-board data handling with partial redundancy between tasks. The ACS algorithm is integrated in the software on one of the computers and is not redundant on the other one. Attitude control actuation is provided by three per-

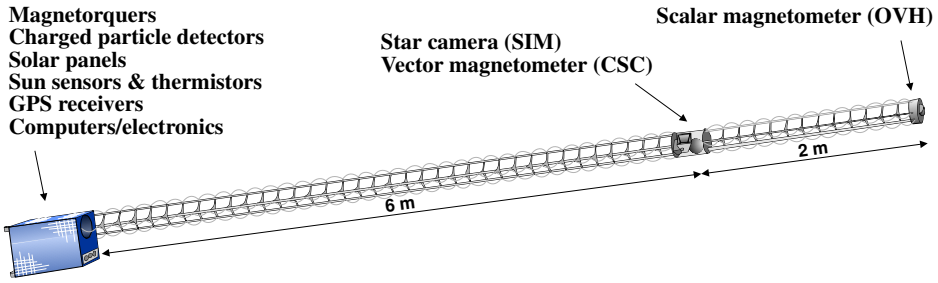


Figure 5.1: *The Ørsted satellite with the boom deployed.*

Table 5.1: *The Ørsted satellite instruments.*

Abbrev.	Instrument	Assignment
CSC	Vector magnetometer	Science instrument that measures the magnetic field in three axes with very high accuracy.
SIM	Star camera	Science instrument that provides very precise inertial attitude estimates based on a star pattern recognition system.
OVH	Scalar magnetometer	Used for in-flight calibration of the vector magnetometer.
CPD	Charged particle detectors	Six science instrument that measure high energy electrons, protons and Alpha particles.
GPS	Global positioning system	Two GPS receivers provide orbit position and satellite velocity estimates.
SSA<i> SSB<i>	Sun sensors, $i \in$(1..8)	Two sets of 8 sun sensors give the direction of the sun for use in attitude determination.
SST<i>	Sun sensor thermistors, $i \in$(1..4)	4 thermistors provide sun sensor temperature measurements for the purpose of thermal compensation.

pendicalur two-redundant electro-magnetic coils, called magnetorquers (MTQs). The MTQ currents are generated with individual coil drivers (CDA<i> and CDB<i> where $i \in$(X,Y,Z)).

The configuration of the ACS software is shown in figure 5.2. The two scientific sensors, CSC and SIM, are the most vital ACS sensors. The initial detumbling entirely relies on the CSC measurement that is used to compute the rate of the satellite with respect to the external magnetic field (this is called B-dot). When the boom is deployed, attitude stabilization is based on the SIM's inertial attitude estimates that are transformed with orbit position information into local attitude and rate estimates. The SIM instrument is not operable if the angular velocity is higher than 10 deg/min or if bright objects are in the field of view of the camera. Consequently, a secondary attitude determination algorithm is required. This is based on an extended Kalman filter (EKF) that computes attitude and rate estimates from the measured magnetic field and sun vector together with model vectors of the geomagnetic field and the sun direction. The sun vector is computed from the sun sensor (SS) measurements where the SS temperature dependency

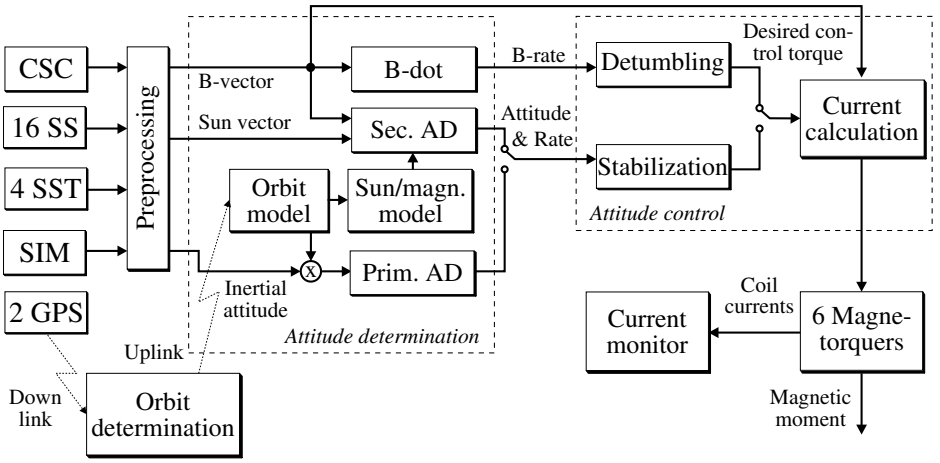


Figure 5.2: *The attitude control system configuration. The main components are sensors, attitude determination, attitude control, and actuators.*

has been compensated using a temperature measurement (SST). The orbit model is dynamically updated from ground every 5th day based on information from the GPSs. The attitude control algorithm determines a desired control torque that is used to compute the coil currents based on the measured geomagnetic field direction. The producible control torque may differ from the desired torque because the physical relationship between the generated torque, the magnetic moment of the magnetorquers, and the external magnetic field is given by a cross product. Finally, a coil current measurement (CCM) provides the absolute value of each magnetorquer current.

The detumbling control cycle runs with a sampling time of one second. In each cycle, the magnetorquer currents are activated for about 900 ms and paused in the remaining period. The magnetic influence from the coils on the CSC, when it is stowed inside the body, prevents the usage of the instrument during the active period. A measurement of the external magnetic field is therefore acquired during the zero-current period. After boom deployment the stabilization controller runs every 10th second. The magnetorquer currents can now be active all the time because the magnetic influence on the CSC in the boom-out configuration is smaller than 15 nT (below 1 per mille of the geomagnetic field amplitude). This small disturbance is without importance for attitude control and it can be compensated in the scientific measurements.

A general introduction to attitude determination and control of the Ørsted satellite is provided in Bøgh *et al.* (1997). Details on attitude determination using a Kalman filter is given in Bak (1996) and details on attitude control using magnetic actuation is given in Wisniewski (1996).

5.2.1 Requirements for Autonomy

The Ørsted satellite is controlled from a single ground station situated in Denmark permitting only 2-3 contact periods of 10 minutes each 24 hours. Therefore, autonomous operation and on-board fault handling is required to avoid serious impact on the scientific mission or even temporal loss of the satellite. Some of the functionality that is traditionally implemented on ground stations must, consequently, be moved to the space segment.

To illustrate the necessity of fault handling consider the consequence of a CD shunt resistor short circuit during detumbling, shown in figure 5.3. The purpose of detumbling

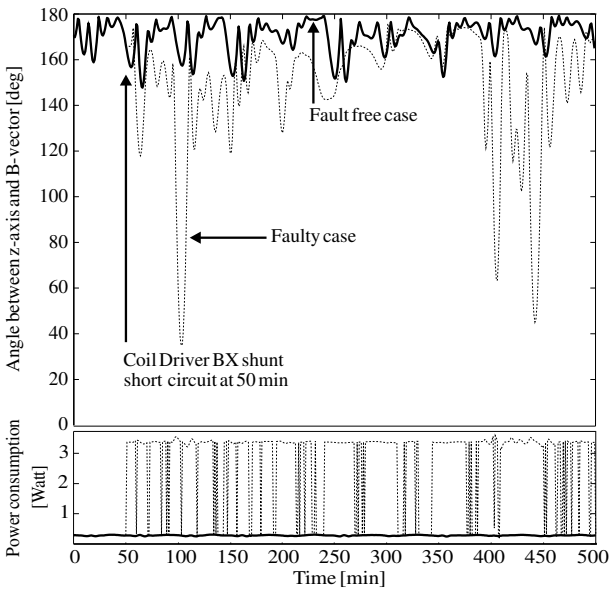


Figure 5.3: *Example on a hardware fault occurring during detumbling of the satellite. The control error increases significantly and the average power consumption becomes very large.*

is to align the satellite's z-axis with the local geomagnetic field vector (an angle of 180 degrees). The fault causes the control error to increase from a peak value of about 30 degrees to 140 degrees. This reduces the availability of high rate communication with ground. Even worse is the consequence on power consumption that increases from about 1/4 Watt to about 3 Watts in average. This is three times the allocated power for ACS, so the consequence is that the batteries will be drained and subsequently the ACS or some on-board instruments will be disengaged.

The basic requirement for the FTCS is to keep running as far as possible in case

of single faults, and make a close-down if continued operation is impossible. The fundamental principle is that it is more sensible to leave the satellite without control than produce erroneous torques, because the system is conservative.

A summary of requirements for fault tolerance together with requirements for commands and monitoring are given in the following list:

- *Fault handling*. Any single fault in CSC, SIM, SSs, SSTs, CDs, and associated transmission lines shall be handled if it is a potential risk for the mission. Fault accommodation can be done through reconfiguration between redundant devices, graceful degradation, or close-down. GPS receiver faults shall not be handled on-board.

- *Sampling time monitoring*. Critical deviation in the control cycle sampling time shall be handled.

- *Operational phase commands*. The handling of the detumbling and stabilization phases is controlled from ground. It shall be possible to activate the algorithms by ground command.

- *ACS execution control commands*. It shall be possible to activate and deactivate the entire control cycle by ground command. It shall also be possible to disable and enable only the torque generation, but continue attitude determination.

- *Attitude determination mode handling*. The switching between the primary and secondary attitude determination algorithms shall be handled on-board. When the primary mode is re-enabled after a period with secondary mode, the controller shall be disabled for two minutes to allow the angular velocity estimate to converge.

- *Attitude control mode handling*. The stabilization controller has three modes: Low power mode, high power mode, and boom-down mode. In low power mode the control torque is restricted to minimize magnetic disturbance on CSC. This is feasible for small attitude corrections, but for large manoeuvres high power mode is required. The switching between these two modes shall be handled on-board. If the boom points towards Earth, control shall be disabled and a message shall be sent to ground. The boom-down controller can then be activated by ground command. When the boom points upwards again, the normal high power mode shall be activated automatically.

- *Eclipse monitoring*. The use of sun sensors in secondary attitude determination requires monitoring of periods with sun and eclipse.

- *On-board autonomy management*. It shall be possible to disable/enable the on-board autonomous operations.

- *Reset command*. A ground command shall be able to restore all internal variables in ACS to default values.
- *Telemetry*. Status information shall be sent to ground following any ACS reconfiguration. Housekeeping information with key variables shall be sent every minute.
- *Command verification*. All received commands shall be acknowledged with one of (received, completed) and one of (success, illegal command).

These requirements are all of a general character. Detailed specification of the requirements for the ACS supervisor fault handling will be available after an analysis of the failure modes of the instruments and an examination of the behaviour of the control system in case of faults.

5.3 Attitude Control System Analysis

This section provides a list of possible failure modes of the ACS instruments and an analysis of their propagation into end-effects on the attitude control performance. An overview of possibilities for reconfiguration is also given. The analysis is used later for fault detection and isolation design and for specification of the decision logic requirements.

5.3.1 Failure Modes

The assessment of potential failure modes is supported by a *physical structure model* as introduced in section 3.2.1 about fault modelling. The level of detail in the physical structure depends for the individual subsystems on the á priori knowledge on failure modes and the possibilities of reconfiguration. A short description of each subsystem and the identified failure modes are given below.

Vector magnetometer (CSC). The magnetic field is determined independently in three axes (X, Y, and Z) with individual fluxgates and converted with a common ADC. Potential faults are a broken wire causing a reading to be zero or a malfunction in the amplifier circuit causing a reading of ± 65535 nT. The zero fault can also happen simultaneously in all three axes due to ADC failure or a disconnected wire. The CSC sample is sent to ACS from the other computer via a bus with non-deterministic delay so temporarily missing samples are possible.

Star camera (SIM). The SIM has its own computer running an algorithm that searches for matching patterns in a star catalog, so any fault in the SIM will cause an absence of samples. This means that all samples received by ACS are assumed adequate

for attitude control. A SIM fault will thus be equivalent to a blackout due to bright objects in the field of view or high angular velocity of the satellite, so there is no dedicated failure modes for this instrument.

Sun sensors (SS). Each of the 8 SS heads consists of two small solar cells that each generates a current depending on the amount of illumination. The two cells are electrically independent. All currents are converted to voltages with individual amplifiers and sampled with a common ADC through a multiplexer. Potential faults include broken wires or defects in the semiconductor that lead to zero voltage reading (corresponding to zero current). Amplifier malfunctions may lead to maximum voltage reading (corresponding to a current of 2.0 mA). All readings can fail to zero simultaneously due to broken wires or ADC failure.

Sun sensor thermistors (SST). Each SST consists of a temperature sensitive resistor mounted in a Wheatstone bridge that provides a voltage reading. The voltage is read by an ADC through a multiplexer. Potential faults are broken wires, disconnected resistor elements, and ADC faults that lead to zero voltage reading (corresponding to a temperature of about 72 degrees centigrade) or amplifier faults that lead to maximum voltage reading (corresponding to a temperature of about -21 degrees centigrade).

Coil drivers (CD). The current in each of the six magnetorquers is controlled by a coil driver as illustrated in figure 5.4. Possible faults in this circuit are the following:

- Shunt resistor short circuit.
- Shunt resistor disconnected.
- Sign logic fails permanent negative.
- Sign logic fails permanent positive.
- Power supply transistor T5 fails open.
- Power supply transistor T5 fails short.
- One of the four bridge transistors (T1-T4) fails short.
- One of the four bridge transistors (T1-T4) fails open.

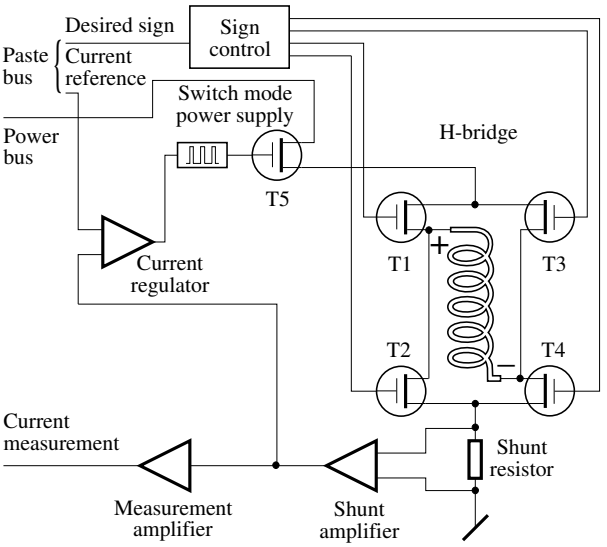


Figure 5.4: *Electrical diagram of a coil driver and magnetorquer in the Ørsted satellite. The magnetorquer is driven by a bridge coupling of four power transistors. The bridge is controlled to provide the desired direction of current flow by the sign control block. The desired current amplitude is controlled by a regulator with feedback from a shunt measurement. Power to the bridge is provided by a switch mode supply. The absolute current measurement is also made available for the control computer..*

Magnetorquers (MTQ). Each of the six magnetorquers are connected to their coil driver with a positive and negative terminal. Possible faults are:

- Coil disconnects or wiring fails open.
- Coil fails short.
- Coil positive terminal short circuit to ground.
- Coil negative terminal short circuit to ground.

Data busses and wiring. The acquisition of data from sun sensors (SSs), sun sensor thermistors (SSTs), and coil current measurements (CCMs) are requested via a dedicated serial-parallel bus (called Paste) and sampled via multiplexers and an ADC. Transmission faults and conversion faults are reported by the low level driver software.

This list does not cover all possible faults, but is representable for the range of possible end-effects.

5.3.2 Fault Propagation

The end-effects on attitude control performance for the above faults are very difficult to derive by analytical examination using logic models of the fault propagation through each subsystem as suggested in section 3.2.2. The reason is that the dynamic behaviour of the satellite is far too complex for the qualitative analysis. The torque produced on the satellite depends on both the satellite attitude and the rotating external magnetic field. This means that the behaviour of the satellite, following specific failure modes of the magnetorquer current, is not well defined, although an experienced spacecraft engineer may be able to predict the fault's consequence without further analysis. In this thesis the end-effects have been estimated, as an alternative, using a numerical simulation program developed for the attitude determination and controller design.

The simulation program does not include the electrical details of the coil drivers and magnetorquers as presented in figure 5.4. The failure modes of these subsystems are therefore determined by the suggested fault propagation analysis (FPA) using logic models. This case study is presented below and is succeeded by a summary of the final end-effects on attitude control performance caused by the failure modes of the previous section.

5.3.2.1 Fault Propagation of the Coil Drivers and Magnetourqers

The fault propagation analyses of the CD and MTQ are included in the presentation because they demonstrate additional interesting aspects about the problem of modelling feedback loops. Feedback via sensors in a control loop was earlier discussed in section 3.2.2 and illustrated on the benchmark problem in section 3.3.3. The interconnection between the CD and the MTQ illustrates another situation, namely an equilibrium between a current generator and the resistance of the load. The magnetic moment naturally depends on the current output from the supply, but the current flow in the supply also depends on the coil. If for example the coil is broken open circuit, there will be zero current flow. This is conveniently modelled as a bidirectional propagation of discrepancies, which is similar to feedback.

The components of the electrical diagram in figure 5.4 are organized into a set of logic models, shown in figure 5.5. This figure shows the interface variables between the different logic models. The six logic models are listed in table 5.2 through table 5.6.

Table 5.2: *Logic model for the sign control and shunt resistor.*

Sign control	Input	Output
Fault	SignDes	Sign
NoFault	Plus	Plus
NoFault	Minus	Minus
SignFailedPos	Minus	WrongSign
SignFailedNeg	Plus	WrongSign

Shunt resistor	Input	Output
Fault	ShuntCur	ShuntVolt
NoFault	OK	OK
NoFault	Zero	Zero
NoFault	Max	Max
ShuntDisc	–	Max
ShuntShort	–	Zero

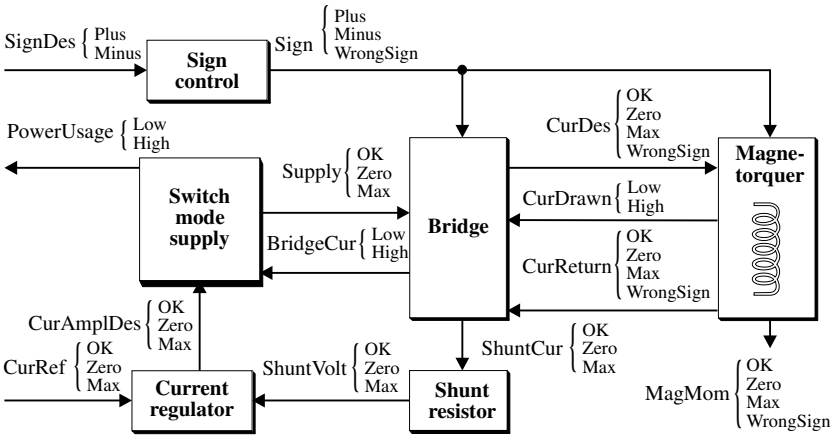


Figure 5.5: *The structure between the logic models of the Ørsted coil driver and magnetorquer circuit in figure 5.4. The figure shows the involved variables and their logic values.*

Table 5.3: *Logic model for the current regulator.*

Current regulator	Inputs		Output
Fault in loop	ShuntVolt	CurRef	CurAmplDes
No	–	OK	OK
No	–	Zero	Zero
No	–	Max	Max
Yes	OK	–	OK
Yes	Zero	–	Max
Yes	Max	–	Zero

Table 5.4: *Logic model for the switch mode supply.*

Switch mode supply	Inputs		Output	
Fault	BridgeCur	CurAmplDes	Supply	PowerUsage
NoFault	–	OK	OK	
NoFault	–	Zero	Zero	
NoFault	–	Max	Max	
–	Low	–		Low
–	High	–		High
T5Short	–	–	Max	
T5Open	–	–	Zero	

Table 5.5: *Logic model for the bridge coupling. The bridge transistor faults are represented here by T1 and T2 only. The other transistors, T3 and T4, have equivalent effects.*

Bridge	Inputs				Outputs		
Fault	Supply	Sign	CurDrawn	CurReturn	CurDes	BridgeCur	ShuntCur
NoFault	OK	Plus	–	–	OK		
NoFault	OK	Minus	–	–	OK		
NoFault	Zero	Plus	–	–	Zero		
NoFault	Zero	Minus	–	–	Zero		
NoFault	Max	Plus	–	–	Max		
NoFault	Max	Minus	–	–	Max		
NoFault	–	–	Low	–		Low	
NoFault	–	–	High	–		High	
NoFault	–	–	–	OK			OK
NoFault	–	–	–	Zero			Zero
NoFault	–	–	–	Max			Max
NoFault	–	–	–	WrongSign			OK
NoFault	–	WrongSign	–	–	WrongSign		OK
T1Short	OK	Minus	–	–	Zero	Low	OK
T1Short	Zero	Minus	–	–	Zero	Low	Zero
T1Short	Max	Minus	–	–	Zero	High	Max
T2Short	OK	Plus	–	–	Zero	Low	OK
T2Short	Zero	Plus	–	–	Zero	Low	Zero
T2Short	Max	Plus	–	–	Zero	High	Max
T1Open	–	Plus	–	–	Zero	Low	Zero
T2Open	–	Minus	–	–	Zero	Low	Zero

Table 5.6: *Logic model for the magnetorquer.*

Magnetorquer	Inputs		Outputs		
Fault	Sign	CurDes	MagMom	CurDrawn	CurReturn
NoFault	–	OK	OK	Low	OK
NoFault	–	Zero	Zero	Low	Zero
NoFault	–	Max	Max	High	Max
NoFault	–	WrongSign	WrongSign	Low	WrongSign
CoilShort	–	OK	Zero	Low	OK
CoilShort	–	Zero	Zero	Low	Zero
CoilShort	–	Max	Zero	High	Max
CoilDisc	–	–	Zero	Low	Zero
PosTermShortGND	Plus	OK	Zero	Low	Zero
PosTermShortGND	Plus	Zero	Zero	Low	Zero
PosTermShortGND	Plus	Max	Zero	High	Zero
PosTermShortGND	Minus	OK	OK	Low	Zero
PosTermShortGND	Minus	Zero	Zero	Low	Zero
PosTermShortGND	Minus	Max	Max	High	Zero
NegTermShortGND	Minus	OK	Zero	Low	Zero
NegTermShortGND	Minus	Zero	Zero	Low	Zero
NegTermShortGND	Minus	Max	Zero	High	Zero
NegTermShortGND	Plus	OK	OK	Low	Zero
NegTermShortGND	Plus	Zero	Zero	Low	Zero
NegTermShortGND	Plus	Max	Max	High	Zero

Inputs to the suite of logic models are current reference, desired sign, and component faults. Outputs are the end-effects on magnetic moment and power consumption.

The characterization of the discrepancies are a three-tuple for unidirectional signals (OK, Zero, Max) and a four-tuple for bidirectional signals (OK, Zero, Max, WrongSign). "Zero" means that the signal failed permanent zero, "Max" means that it failed too high, and "WrongSign" means that it failed with opposite direction. The non-faulty operation (OK) is included because the WrongSign fault in bidirectional variables propagates into an OK state of the unidirectional variables. The reason for this is that the current measurement gives only the current amplitude and not the direction of flow. Finally, the variables associated to the power consumption take the values Low or High describing whether they are lower than normal or OK ("Low") or too high ("High").

The following comments on the logic models are useful to understand the interplay between the components:

- *Magnetorquer terminal faults.* The fault effects for the short circuit of the negative and positive terminals to ground (Pos/NegTermShortGND) are rather intricate because these faults relate to the physical components (terminals) whereas the discrepancies relate to the functional information flow (current and magnetic moment). The problem is that the bridge coupling changes the functional configuration when it changes current direction, so the effects of component faults depend on the current direction (Sign).

- *Current control loop faults.* All the considered component faults except CoilShort, SignFailedPos, and SignFailedNeg effect the current feedback loop. The three excepted faults do not effect the current measurements (they lead to ShuntCur=OK), so they are not included as "Fault in loop" in the model for the current regulator in table 5.3. This is an interesting observation because the effected variables are located inside the feedback loop.

The fault propagation graph (FPG) of the logic models is analyzed in Beologic™ and the rules can be found in appendix B.1. Table 5.7 presents an examination of the relations between faults and end-effects. The last column is added to show the cases where the current measurement can be used for fault detection. The current measurement is wrong if the shunt voltage is faulty (ShuntVolt$\in$[Zero, Max]). It can be seen that a coil short circuit, a sign failure on the magnetic moment, and bridge transistors short circuit faults cannot be detected with the current measurement. A wrong direction of the control torque is catastrophic, so the fact that this is not detectable with the current measurement will show up later to necessitate implementation of a rather elaborate fault detection algorithm.

Table 5.7: *A table showing the relations between faults and end-effects of the Ørsted coil driver and magnetorquer subsystems. The current reference input is forced to the non-faulty value CurRef=OK in this analysis to indicate that only CD and MTQ faults are analysed.*

↓ Fault - End-effect →	Power=High	MagMom=Zero	MagMom=Max	MagMom=WrongSign	CurrentMeas=Wrong
ShuntShort	x		x		x
ShuntDisc		x			x
T5Short		x			x
T5Open	x		x		x
SignFailedPos				x	
SignFailedNeg				x	
CoilShort		x			
CoilDisc		x			x
PosTermShortGND & Sign=Plus	x	x			x
PosTermShortGND & Sign=Minus	x		x		x
NegTermShortGND & Sign=Plus	x		x		x
NegTermShortGND & Sign=Minus	x	x			x
T1Short & Sign=Minus		x			
T2Short & Sign=Plus		x			
T1Open & Sign=Plus		x			x
T2Open & Sign=Minus		x			x

5.3.2.2 End-Effects on Attitude Control Performance

An analysis of the effects on attitude control performance from faults in all subsystems (including coil drivers) can now be performed by simulating the individual component failure modes using a simulation program of the satellite dynamics. The end-effects on attitude control performance are categorized into three classes: small increase in attitude error, large increase in attitude error, and random motion. Also a large increase in power consumption is included as an end-effect. The complete relationship between faults and end-effects on top level is listed in table 5.8. This FPG will be used later in section 5.6 for a completeness check of the fault handling decision logic design. The Beologic™ rules of the complete FPG are listed in appendix B.2.

5.3.3 Severity Assessment

A *severity assessment* on the end-effects yields a requirement for handling of any end-effect except SmallError. The SmallError end-effect will still be accommodated because the same failure modes that lead to this effect, cause more severe end-effects for other coil drivers. These more severe cases shall be handled, so the SmallError end-effect is automatically accommodated.

Table 5.8: *A table showing the relation between faults and end-effects of the Ørsted attitude control system in the two operational phases; detumbling and stabilization.*

		Detumbling				Stabilization			
Instrument	↓ Fault - End-effect →	RandomMotion	LargeError	SmallError	Power=High	RandomMotion	LargeError	SmallError	Power=High
CSC	Any single axis failed zero		x			x			
	Any single axis failed maximum	x				x			
	All axes fails zero	x				x			
	Absence of samples	x				x			
SIM	Absence of samples	-	-	-	-	x			
SS	Any active sensor failed zero	-	-	-	-	x			
	Any active sensor failed high	-	-	-	-		x		
SST	Any active sensor failed zero or high	-	-	-	-		x		
CD	Zero magnetic moment in CDAX or CDBX			x			x		
	Zero magnetic moment in CDAY or CDBY		x				x		
	Zero magnetic moment in CDAZ or CDBZ			x			x		
	Maximum mag. mom. in CDAX or CDBX		x			x			
	Maximum mag. mom. in CDAY or CDBY	x				x			
	Maximum mag. mom. in CDAZ or CDBZ		x			x			
	Wrong sign on mag. mom. in CDAX or CDBX			x		x			
	Wrong sign on mag. mom. in CDAY or CDBY			x		x			
	Wrong sign on mag. mom. in CDAZ or CDBZ	x				x			
	High power consumption in CD				x				x

5.3.4 Remedial Actions

The possibilities for reconfiguration of the ACS for fault accommodation is rather limited because the level of hardware redundancy is very low and the available computational capacity is restricted to execute FDIR algorithms of only modest complexity. The following remedial actions are available:

- *Attitude acquisition*. The acquisition of attitude estimates can be switched between the primary (SIM based) and the secondary (CSC and SS based) attitude determination algorithms.

- *Sun sensors*. Each of the 8 sun sensor heads has two-redundant cells. If a primary head fails, the secondary head can be sampled instead. The secondary attitude determination algorithm is furthermore able to operate without the sun vector input.

- *Coil drivers*. Each coil driver and magnetorquer is two-redundant. Both sets are normally enabled to facilitate maximum control torque. If one coil driver or magnetorquer fails, it can be disabled independently of the other set. The six coil drivers are grouped in two sets (A and B). Each set is powered separately and can be disabled and enabled individually. The power supplies are disabled to save energy if the three coils in one set are not used.

- *Close down*. If a serious fault cannot be accommodated the ACS is closed down and the magnetorquer currents set to zero.

The ACS block diagram from figure 5.2 is repeated in figure 5.6 with extensions for reconfiguration. The possibilities for reconfiguration are indicated either with switches that control the data flow, or with dashed boxes showing groups of modules that can be enabled and disabled. The figure shows the reconfigurable components for both fault handling and operational control. The latter will be covered later.

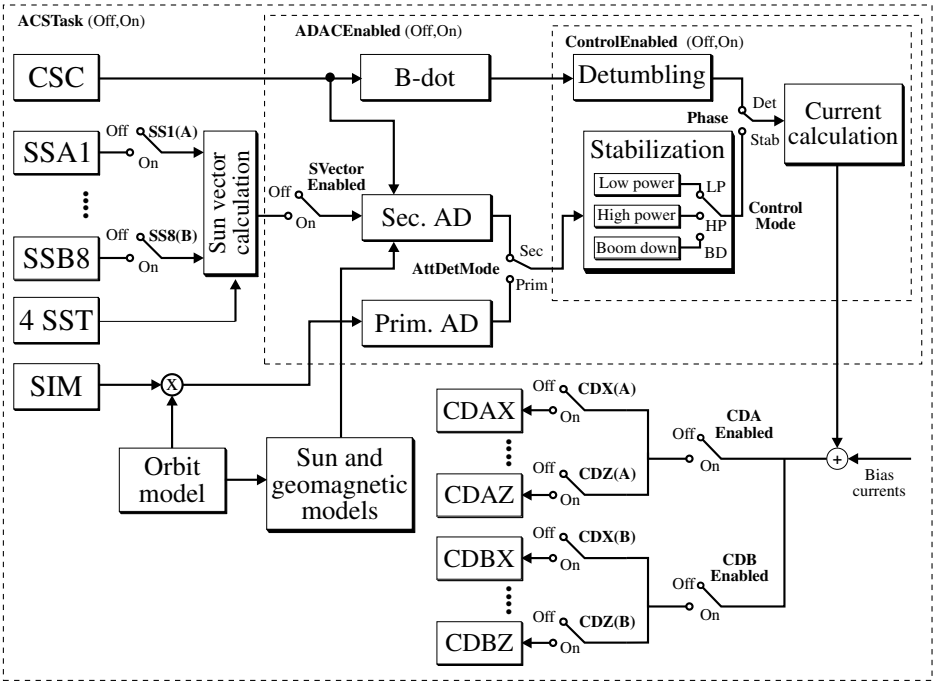


Figure 5.6: *Possibilities of reconfiguration of the attitude control system. Switches indicate the usage of instruments and selection between operational modes. Dashed boxes indicate execution control over groups of functionalities.*

The general *reconfiguration requirement* is that the attitude in the stabilization phase shall change with no more than 10 degrees in pitch or roll and 20 degrees in yaw before reconfiguration is accomplished. The requirements for the detumbling phase are a maximum change in angular rate of 80 deg/min and a maximum change in control error of 60 degrees (2 times the normal variations). An additional requirement is that an increase in power consumption following a fault shall add up to no more than 3 Watt-hours (12 Watt in 15 minutes).

5.4 Fault Detection and Isolation Design

This section gives a short introduction to the methods chosen to detect the occurrence of the faults listed in section 5.3.1. In summary the requirements for fault detection and isolation (FDI) are the following:

- Detection of any faults in CSC. Isolation between the three axes is not required.
- Detection of periods with and without SIM samples.
- Detection of faults in any SS. Isolation between the individual elements is required.
- Detection of faults in any SST. Isolation not required.
- Detection of faults in any CD/MTQ. Isolation between the individual drivers is required.

5.4.1 Overview of Fault Detectors for ACS

The detection of erroneous signals is performed on two levels. The control level includes protection against out-of-limit values on individual channels and also performs outlier filtering on most signals. The FDI algorithms on the next level (which are in focus in this thesis) use the correlation between more signals to detect and isolate faults. The following list explains, for each instrument, the preprocessing performed on the control level and presents the additional FDI methods used on the detector level.

Vector magnetometer faults (CSC). The preprocessing in the control level of each channel includes outlier filtering and out-of-limit test from the range [-65000, +65000] nT. The operational range is approximately within [21000, 46000] nT. If a CSC sample arrives later than 300 ms after the request or if an out-of-limit was detected, the control loop is suppressed. The range check is not performed on the CSC measurement for the non-zero current period in the detumbling phase, because the instrument may be saturated.

The detector level includes, both during detumbling and stabilization, an out-of-limit test on the magnitude of the measured external magnetic field filtered through a first order low pass filter. An out-of-limit is detected if the magnitude exceeds the interval [15000, 55000] nT.

During the stabilization phase, output from the geomagnetic field model is used to verify the magnitude of the magnetic field measurement. A residual is generated as a first order filtered version of the difference between the two magnitudes. The test threshold is set to 1000 nT.

Star camera faults (SIM). The SIM samples are declared absent if the age of the last received sample is older than 15 seconds (nominal sampling rate is 1.2 seconds). The usage of SIM samples is resumed when the sample rate is sufficiently high. This is implemented as a hypothesis test of the sampling rate being higher than 20 seconds.

Sun sensor faults (SS). The preprocessing in the control level of each of the 8 channels includes outlier filtering, out-of-limit test from the range [-0.02, 2.0] mA, and temperature compensation. The operating range for each sensor is approximately [0.0, 1.6] mA. The three elements of the sun vector are then computed from the three sensors with the largest excitation.

Fault detection is achieved by an out-of-limit test on the magnitude of the sun vector. The normal magnitude is 1.0 so a fault is detected if it exceeds the range [0.8, 1.2]. Isolation of the faulty sensor is achieved by inspecting the innovations from the Kalman filter (EKF) of the secondary attitude determination (introduced on page 106). The EKF innovations are scaled relatively to the normalized sun vector. If one of the three innovations exceeds 0.2 (20%) then the corresponding axis is assumed faulty. The SS element that was used as input for this axis is then determined by index information from the sun vector calculation in the control level.

Sun sensor thermistor faults (SST). The preprocessing in the control level of each of the 4 channels includes outlier filtering and out-of-limit test from the range [-20, 60] degrees centigrade. The normal operating range is [-15, 45] degrees centigrade. There is no need for further fault detection as both failure modes fall outside this range.

Coil driver and magnetorquer faults (CD/MTQ). Detection of faults in CDs and MTQs are primarily performed by comparing the current measurement with the desired current. This is not sufficient, as revealed in section 5.3.2, because the current measurement does not provide directional information and is not effected by a coil short circuit.

In detumbling, the coil currents are measured twice each sample, once in the zero-current period and once in the active period. The zero-current measurement is used to assure that the corresponding magnetic field measurement is not disturbed by the magnetorquers. In the stabilization phase, only one measurement is taken. The preprocessing of the zero current measurement includes outlier filtering and out-of-limit check whereas the other measurement only has an out-of-limit test. The test range is [-8, 558] mA where the normal operating range is [0, 550] mA.

Non-zero currents in the detumbling zero-current period is detected by a fixed threshold for each individual CD/MTQ. The test threshold is 10 mA corresponding to a 1000 nT disturbance on CSC.

The primary detection of faults in CD/MTQ, when currents are non-zero, is done by comparing the absolute value of the desired currents with the current measurements. The difference is filtered with a first order filter and compared to a threshold of 50 mA.

The problem with detection of wrong direction on currents, coil short circuits, and bridge transistor short circuits (see table 5.7) can be solved for the detumbling phase, where the coupling between the magnetorquer currents and the magnetic field measurement from the CSC can be utilized. Estimates of the coil currents are calculated from the magnetic disturbance and compared to the desired currents. In this way, the failed axis is determined. Isolation between the two CD/MTQs in the identified axis is achieved by comparing the currents in each CD/MTQ to the residual. It is important to notice that the magnetic coupling in the stabilization phase is not strong enough to make this FDI method feasible.

It is a requirement for all the fault detectors, that use outlier filtered signals, to avoid false detections during the initialization of the filters. This is achieved by postponing the activation of the FDI algorithms a few samples following an ACS activation command.

These FDI algorithms are able to catch all the required component faults, except the CD/MTQ faults that cannot be detected by the current measurement during the stabilization phase (see table 5.7). *These faults cannot be detected by simple means, so the requirements cannot be fulfilled on this point !* The wrong direction error could be detected by a small extension to the hardware in the power drive that provides a signature on the current measurement. The remaining faults could be detected by analytical redundancy methods, where the real torques are estimated from the angular acceleration of the satellite and compared to the expected torques. This requires extensive non-linear modelling and has not been feasible for implementation on the 80186 platform with the present load of the application software.

5.4.2 Assessment on Potential False Alarms

An inspection of the dependency between the functional model of the ACS and the FDI algorithms yields important information about potential false alarms and erroneous isolation of faults:

- The CSC is very sensitive to the magnetorquer currents during the detumbling phase. This means that the CSC can erroneously be declared faulty if a CD/MTQ fault causes a non-zero current during the zero-current period.
- The CSC measurement is used during the detumbling phase to detect CD/MTQ faults. This means that a CSC fault may cause false detection of CD/MTQ faults.
- The CSC fault detector depends on the on-board geomagnetic field model to compute the magnitude of the estimated magnetic field. The magnetic field model determines the local field vector from the orbit position given by the on-board orbit model. Correct orbit position estimates requires updates every 5th day with valid orbit data. If the orbit model update fails then the estimated magnetic field magnitude will differ from the real value and the CSC instrument may be incorrectly declared faulty.

- The on-board eclipse monitor uses the magnitude of the sun vector measurement to identify sunrise and the orbit position estimate to predict sunset a few minutes before the sun disappears. This monitor is thus sensitive to both orbit model malfunctions and sun sensor faults.

These dependencies must be taken into consideration in the subsequent decision logic design and realised during the verification of the FTCS behaviour.

5.5 Supervisor Decision Logic Design

The prerequisites for the decision logic design have now been developed and are given by the general requirements from section 5.2.1, the detected and isolated fault events from section 5.4, and the list of remedial actions from section 5.3.4. This section treats the design of the decision logic that handle both these fault events and also the operational command/monitoring interface. The decision logic is implemented as state-event machines (SEMs) and analyzed in the Beologic VisualState™ tool-kit The usage of VisualState™ imposes a few restrictions on the design. The principles of the analysis are, anyway, generally applicable. The purpose of this section to introduce solutions to some typical problems encountered in the design of the ACS supervisor. An overview of the SEMs is first given whereupon examples are drawn to illustrate key problems and solutions.

5.5.1 Overview of the Decision Logic State-Event Machines

The inference of the SEMs, implemented in the decision logic module, makes a logic mapping from inputs (commands, faults and monitoring events) to outputs (monitoring information and remedial actions). The remedial actions are represented as changes in the active states of the SEMs, so control level updates are executed by transferring the new states after an inference to the control level. The names of the SEMs are therefore identical to the switches in figure 5.6. Not all decision logic tasks are adequately solved by SEM logic, so the decision logic module supports the SEM inference mechanism with input preprocessing, output postprocessing and some utility functions (e.g. timer functions). The present design of the SEMs allows only one active input in one call, so the decision logic task scans the list of active events and calls the inference mechanism in each iteration. Multiple events can occur in one sampling instant because more detectors can be active on the same time. This is the background for the SEM design that is presented in this section.

The choice of the number of SEMs and their mutual dependencies are important in the design of the decision logic so simplicity can be ensured. The Ørsted ACS decision logic has not been designed with any formal approaches, but with the common philosophy that each SEM shall represent an independent functionality as far as possible. An

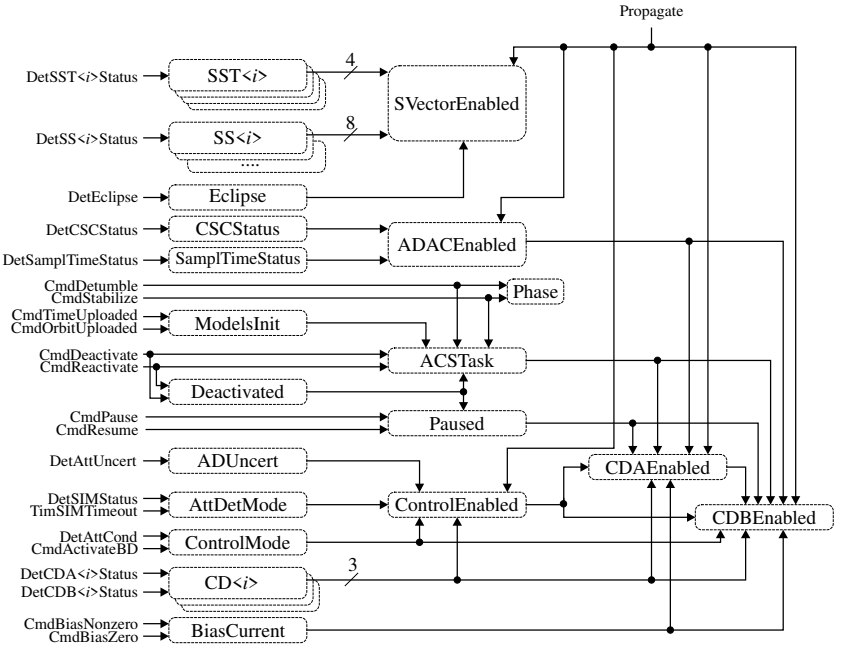


Figure 5.7: *The internal structure of the Ørsted ACS decision logic's state-event machines. Inputs to the state-event machines are commands, events from detectors, time-out signals from external timers, and internal dependencies from other state-event machines. The input "Propagate" is used to trigger transition of the state-event machines that have internal dependencies. The CmdReset command is not shown as well as all the dependencies from the CmdDetumble and CmdStabilize commands.*

overview of the SEMs is presented in figure 5.7 and the tables 5.9-5.10 contain lists of inputs, outputs, and assignments of SEMs. These tables include items in connection to operational mode handling that has not been treated so far. Their assignments are clear from the explanation in the tables. The decision logic is realized with 32 state-event machines with a total of 83 states and 116 rules relating the inputs and old states with new states and outputs.

The overview of the SEMs in figure 5.7 shows the influence from inputs to SEMs and the dependency between the individual SEMs. Each box represents a SEM that can be in one state and change into another state when an input is activated. A dedicated input called "Propagate" is included to trigger the propagation of changes through the chain of dependent SEMs. After the decision logic module has called the inference mechanism with an active input it is then necessary to repeat the call to "Propagate" as long as it remains an active input. It is generally not desirable to solve the problem

Table 5.9: *Inputs to the Ørsted decision logic's state-event machines.*

Input	Explanation
CmdDetumble	Activate detumbling control.
CmdStabilize	Activate stabilization control.
CmdDeactivate	Stop ACS control task.
CmdReactivate	Restart ACS control task in same state as before CmdDeactivate.
CmdPause	Stop attitude control and torque generation but continue attitude acquisition.
CmdResume	Resume attitude control and torque generation.
CmdReset	Reset all internal states to default values at boot-up.
CmdActivateBD	Enable boom-down controller.
CmdBiasNonzero	A nonzero bias to the control currents has been requested.
CmdBiasZero	A zero bias to the control currents has been requested.
CmdTimeUploaded	A time reference has been uploaded from ground. This is required for correct operation of the orbit model.
CmdOrbitUploaded	Orbit parameters have been uploaded. This is required for correct operation of the orbit model.
DetCSCStatus	Failure status of CSC, (Fault, NoFault).
DetSS<i>Status	Failure status of active SS sensor number $i \in$(1..8). The active sensor is either SSA<i> or SSB<i>, (Fault, NoFault).
DetSST<i>Status	Failure status of Sun sensor thermistor number $i \in$(1..4) failed, (Fault, NoFault).
DetCD<s><i>Status	Failure status of coil driver/magnetorquer in axis $i \in$(X, Y, Z) of set $s \in$(A, B), (Fault, NoFault).
DetSIMStatus	Monitoring of SIM, (NotOk, Ok).
DetSamplTimeStatus	Failure status of controller sampling time, (Fault, NoFault).
DetAttCond	Monitoring of attitude deviation from set-point. Three categories represent the limits between low power mode, high power mode, and boom-down (LP, HP, BD).
DetEclipse	Monitoring of periods with sun and eclipse (Sun, Ecl).
DetAttUncert	Monitoring of the convergence of the extended Kalman filter in secondary attitude determination, (NotOk, Ok).
TimSIMTimeout	Time-out signal from the external SIM timer.

Key Cmd : *Telecommands from ground.*
Det : *Fault detector or monitor events.*
Tim : *External timer events.*

Table 5.10: *Outputs from the Ørsted decision logic's state-event machines.*

Output	Explanation
IllegalCommand	The command received is invalid for the present state combination.
BDControlRequest	Request to ground for activation of the boom down control mode.
NotBoomDown	The boom down control mode command CmdActivateBD was sent, but the boom is not down anymore.
StartSimTimer	Start the external timer for re-enabling of attitude control after SIM has become valid again.

Table 5.11: *States in the Ørsted decision logic's state-event machines.*

State event machine	Explanation
Phase	Mission phase (Det=detumbling, Stab=stabilization).
ACSTask	Main ACS control task, (On=enabled, Off=disabled).
ADACEnabled	Switch for attitude determination and control, while control task still running. Disabled if CSC fails or sampling time is violated, (On=enabled, Off=disabled).
ControlEnabled	Switch for attitude control output, while attitude determination still running, (On=enabled, Off=disabled).
Paused	Internal memory for pause-resume commands, (YES=paused, NO=not paused).
Deactivated	Internal memory for Deactivate-Reactivate commands, (YES=deactivated, NO=not deactivated).
AttDetMode	Attitude determination algorithm, (Sec=secondary mode, PrimTimerRunning=primary mode and SIMTimer running, Prim=primary mode and SIMTimer stopped). The output StartSIMTimer is activated when entering PrimTimerRunning, and Prim is entered when the TimSIMTimeout is received. See the TimSIMTimeout for further explanation.
AttUncert	Uncertainty of the attitude estimate computed by the Kalman filter in the secondary attitude determination algorithm, (NotOk, Ok).
ControlMode	Attitude control mode, (LP=low power, HP=high power, BDRequested=high power and request, BD=boom down). When the input DetAttCond=BD is received, BDRequested is entered and a request for boom down control is sent to ground. When the command ActivateBD is received, the state BD is entered.
ModelsInit	Internal memory showing the update status of the time reference and the orbit model (None=no models updated, Time=time reference updated, Orbit=orbit model updated, All=both items updated).
CSCStatus	Internal memory for CSC status, (Fault, OK).
SS<i>	8 Switches for the sun sensor sets, $i \in$(1..8), (AOnBOn, AOnBOff, AOffBOn, AOffBOff).
SST<i>	Internal memory for sun sensor thermistor state, $i \in$(1..4), (On, Off)
SVectorEnabled	Sun vector enabled for secondary attitude determination (On=enabled, Off=disabled)
CD<i>	3 Switches for the coil driver sets in axis $i \in$(X, Y, Z), (AOnBOn, AOnBOff, AOffBOn, AOffBOff)
CD<s>Enabled	Switch for coil driver set $s \in$(A, B), (On, Off)
BiasCurrent	Internal memory that shows whether a nonzero bias current has been requested from ground, (Zero, Nonzero).
SamplTimeStatus	Internal memory for the status of controller sampling time, (Fault, OK).
Eclipse	Internal memory for the eclipse status (InEclipse, InSun).

with internal dependencies with such a propagation-signal, because it restricts the capabilities of consistency checks as will be evident below, but it is a necessary requirement in connection with the VisualState™ software. A more appropriate solution is to include additional variables (which are not *states* of SEMs) that describe the internal dependencies, but this is not feasible in VisualState™. Another alternative is to modify the structure of the SEMs in such a way that each SEM in a dependency chain has direct event input and no SEM dependencies. This solution is undesirable because the SEM conditions become unnecessary complex even for simple applications.

The following examples describe specific details on the design using the state-event technique.

5.5.2 Controller Mode - Example with Internal Memory

The requirements for the controller operational mode handling was given on page 109. The ControlMode SEM uses an internal state (BDRequested) to memorize the situation where the controller is in high power mode and a need for boom down control has been detected and requested (see figure 5.8). The detection of a boom down condition

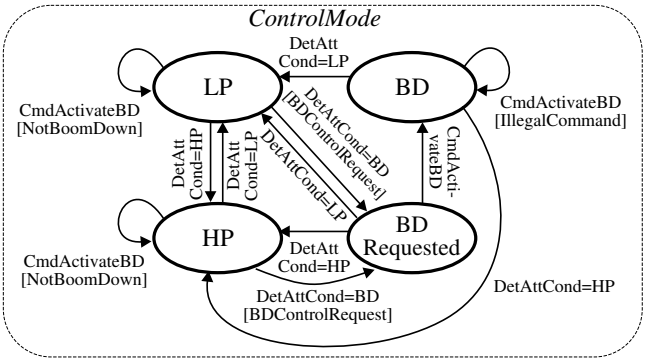


Figure 5.8: *A state-event machine for handling of the attitude controller mode. The controller can be in either low power (LP), high power (HP), or boom down (BD) mode. The fourth state (BDRequested) memorizes when a BDControlRequest has been sent and the SEM is waiting for a CmdActivateBD command. Square brackets indicate outputs.*

leads to a request for boom down control, that is sent to ground, and subsequently the SEM enters into BDRequested. Boom down control mode (BD) is entered when the ground command, CmdActivateBD, is received. If a boom up condition is detected (DetAttCond=LP or HP) while the SEM is waiting for the CmdActivateBD command the corresponding mode is enabled (LP or HP) and when the command arrives, it will be refused and the output, NotBoomDown, will be sent to ground.

5.5.3 Attitude Determination Mode - Example with External Timer

The SEM for handling the two attitude determination modes is shown in figure 5.9. The requirement to disable the controller output (ControlEnabled=Off) during two minutes following a transition to the primary mode is controlled with an additional state that is active while an external timer counts down.

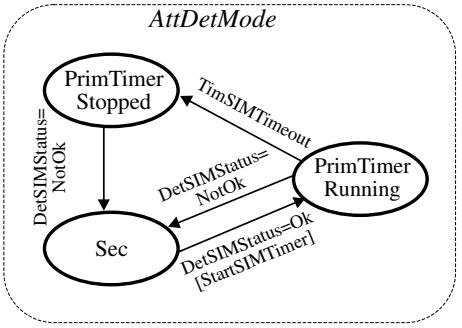


Figure 5.9: *A state-event machine for the attitude determination mode handling. The attitude determination can be in either primary (PrimTimerStopped) or secondary (Sec) mode. The extra state PrimTimerRunning is used to memorize that the external SIMTimer has been activated and the SEM is waiting for a timeout signal.*

5.5.4 Controller Enabled Switch - Example on Internal Dependencies

The ControlEnabled SEM illustrates how internal dependencies between SEMs are used (see figure 5.10). The transitions to On and Off are triggered with the Propagate input and determined by a combination of the current states of other SEMs.

5.5.5 How to avoid Combinatorial Explosions

It is important to separate naturally independent functionalities into individual state-event machines. Otherwise, combinations of these functionalities will be assigned superfluous states leading to excess complexity. An alternative design of the ControlMode SEM is given as an example. It would have been intuitively reasonable to let this SEM handle all controller modes, which include the four states seen in figure 5.8 but also the pause and detumbling modes. This leads to the fairly complex SEM shown in figure 5.11. It is not always obvious how to define the set of SEMs, but care should be taken to avoid these combinatorial explosions. A formalized approach to this problem is given in Zamanabadi *et al.* (1996) using the concept of *extended* state-event machines.

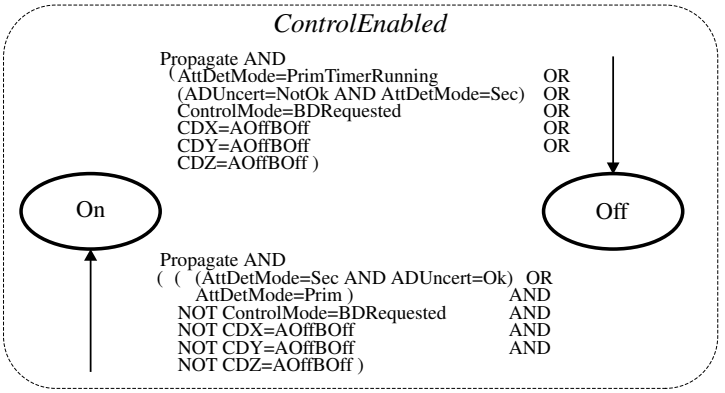


Figure 5.10: *A state-event machine to control the execution of the attitude controller algorithm. The controller is disabled in a number of different situations that is represented by a combination of other states.*

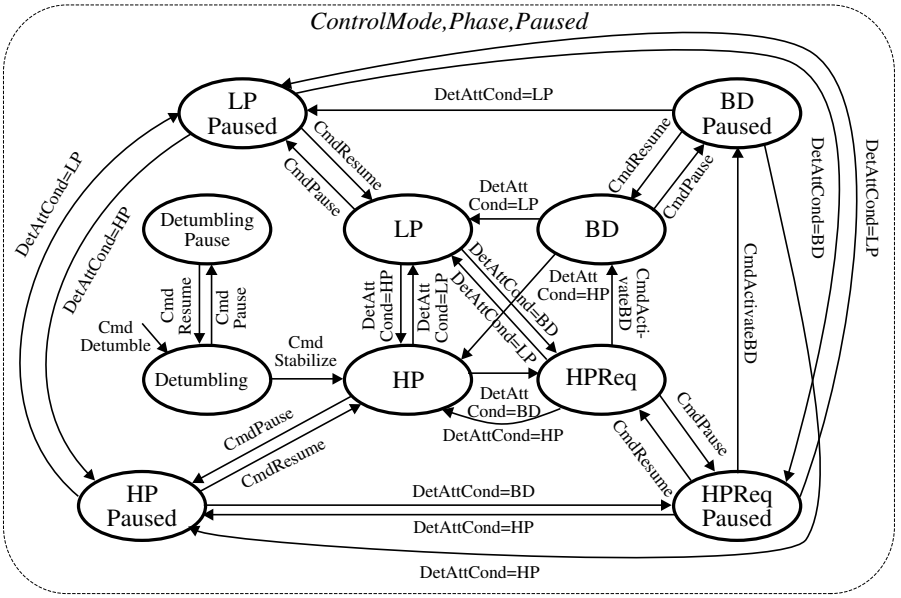


Figure 5.11: *Illustration of an inappropriate SEM design. The ControlMode SEM from figure 5.8 has been combined with the Pause and the Phase SEMs and the result is an unnecessary complicated design.*

5.6 Decision Logic Verification

The purpose of this section is to examine the properties of the SEM design that can be tested before implementation and illustrate these properties with a few examples.

The verification for correctness is performed in three ways: First, an analysis of the SEMs is performed in VisualState™ to ensure consistency of the inference rules. Secondly, the decision logic's SEMs are combined with a SEM description of the entire system to locate potential dead-ends. Third, correct operation against the fault analysis in section 5.3.2.2 is verified with Beologic™ AIT. The latter requires a translation of the fault reconfiguration part of the VisualState™ rules into AIT rules which is possible although not automatic. The two tool-boxes supplement each other, so the conversion is necessary. The Beologic™ products are not the ultimate tools for this task, but are used here to investigate the principles.

5.6.1 Stand-alone Check of the SEM Rules

The coherence between the decision logic's SEMs is analysed in VisualState™ for a number of properties. The analysis does not guarantee correct operation, but it helps to locate important problems.

Correct initialization. A default state must be assigned to each SEM to ensure a deterministic behaviour after initialization or reset. An error message will be reported if not all SEMs have initial states defined.

Unused inputs, outputs, states, and rules. Unused items represent potential design errors. Inputs, outputs, and states, that have been declared, but are never used indicate incomplete implementation and are pointed out with a warning. Unused rules are more complicated. A rule will never be used, if the condition side is unattainable. This is the case when an illegal combination of states is used as condition. Unused rules cause an error message. No examples are given, but the check is used to find illegal state combinations described below.

Dead-ends. A dead-end is a state that can only be left with a complete reset. A warning is issued if dead-ends exist in single SEMs. It is also possible that the entire system of SEMs gets locked up due to internal dependencies. This situation will cause an error message.

The SS<i> SEMs can be used to illustrate this situation. If both an A sensor and a B sensor have failed, the entire sun vector is rejected until one of the commands, CmdDetumble or CmdStabilize, is received. If these dependencies on the commands are forgotten, the SS<i> AOffBOff-state will be an illegal dead-end.

Dead-ends in single SEMs may be desired by design as it is the case with the ModelsInit SEM. This SEM is used to memorize that the time and orbit parameters have

been uploaded to the on-board models. This information is only lost in case of computer reset, so this SEM has a legal dead-end.

Contradiction between rules. A contradiction exists if two rules with the same conditions (input and internal dependencies) forces a SEM into two different states. The ControlEnabled SEM in figure 5.10 serves as an example. If, for example, the condition "NOT ControlMode=BDRequested" was forgotten in the on-transition, it is possible to have a transition into both the On- and Off-state. This causes a contradiction warning message.

Illegal state combinations. It is possible to utilize the check for unused rules (described above) to verify that state combinations, known by design to be illegal, will never occur. This is achieved by temporarily entering an additional rule that has the illegal state combination as condition. If an error message indicates that the rule will never be activated, then the design is successful.

As an example consider the fact that the coil driver sets shall be Off if the on-board models have not been initialized in the stabilization phase. This can be verified by adding the rule,

```
Propagate AND CDAEnabled=On AND (NOT ModelsInit=All) AND Phase=Stab:IllegalOutput
```

A warning message will indicate that this rule will never be activated. The input (Propagate) and output (Illegal) of this rule are not important, but are used to constitute a legal rule.

It is not possible to verify all illegal combinations in VisualState™ because the propagation of events through the SEMs allow many state combinations to be legal, although they will never appear together after the state propagation is finished. It is, for example, not possible to verify that the coil driver sets are disabled if the CSC has failed. In the first iteration after a DetCSCStatus=Fault input, the coil driver sets can be on. Only after the third iteration both coil driver sets will be turned Off. This problem is specific for VisualState™ and it would be possible to make a complete check with a dedicated software tool.

5.6.2 Combined Check with SEM Model of the Control Level

It is a wrong design if the decision logic disables the control level subsystems that produce inputs for active fault detectors, when it reconfigures in connection with fault handling. It is possible to make an automatic check for such dead-locks in VisualState™. Consider the following example that was discovered and corrected in the Ørsted software at a very late point in the development. The CSC FDI uses both the on-board geomagnetic model and the measured magnetic field for fault detection. If the sensor is detected faulty, the switch ADACEnabled is disabled (see figure 5.6 on page 119).

In the earlier software version this switch also disabled the call to the on-board models and thereby update of the geomagnetic model. Consequently, the CSC instrument will never be declared non-faulty again. This means that an intermittent fault or false alarm will cause the CSC to be permanently disabled and the satellite left uncontrolled (until a ground command restores operation).

The above problem can be analysed by combining the decision logic's SEMs with a SEM description of the functionality of the control level as shown in figure 5.12 for the old software version. Two additional SEMs are introduced: One for the geomagnetic

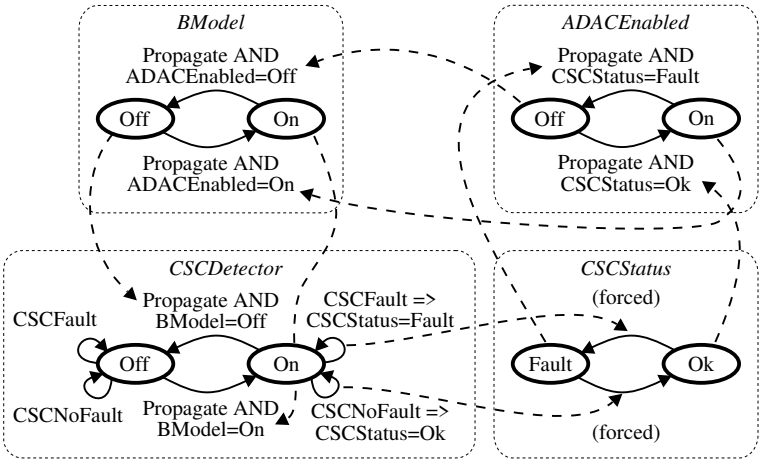


Figure 5.12: *Example on a method to check for possible dead-ends between the supervisor's decision logic and the control level. A SEM description of the control system and FDI (BModel and CSCDetector) is combined with the SEMs of the supervisor's decision logic (ADACEnabled and CSCStatus). The dependencies between the SEMs are shown with dashed lines. In this example, a CSC fault will lead to a dead-end in the system.*

model (BModel) and one for the CSC fault detector (CSCDetector). These are turned On or Off depending on the status of the ADACEnabled SEM. A change in CSC failure status (CSCFault or CSCNoFault) is only propagated through the CSC detector if the detector is on. This is shown as the dotted lines connecting CSCDetector with CSCStatus. These lines represent the input, DetCSCStatus, that carries the CSC status information. The system is now analysed in VisualState™ and a dead-end is found because the CSC fault event turns ADACEnabled Off, which causes the detector to be Off and thereby to be insensitive to the CSC failure status inputs.

Generally, the entire system can be analysed in one step this way, but this is not possible in VisualState™. This tool-kit can only determine total system dead-end or dead-ends in single SEMs. It cannot locate dead-ends in the interaction between more SEMs if there is not a total system dead-end.

5.6.3 Combined Check with Fault Propagation and Reconfiguration Analysis

A completeness check of the supervisor's fault handling can be performed in Beologic™ AIT, where it is verified that the decision logic handles all considered faults. The entire logic model of the system is implemented in AIT following the principle illustrated earlier in figure 3.8 on page 39. This includes the fault propagation analysis from section 5.3.2, the logic behaviour of the fault detectors and the reconfiguration, and the operation of the fault handling part of the decision logic's SEMs. The combined rule base is then examined in the same way as for the benchmark example on page 56. Appendix B.3 contains a representative extract of all the rules that describe the entire ACS logic model.

A simple example illustrating the principle of analysis is given in table 5.12 that lists the rules for analysis of a coil driver sign fault. A wrong sign on the magnetic moment

Table 5.12: *An example on Beologic™ rules used to verify completeness of the supervisor decision logic.*

#	Rule
	Coil driver FPG:
1	`CDAZMagMomWrongSign = (CDAZSignFailedPos or CDAZSignFailedNeg)`
	Propagation through reconfiguration switches:
2	`CDAZMagMomWrongSign_ = (CDAZOn and CDAEnabledOn and CDAZMagMomWrongSign)`
	FPG to end-effects:
3	`RandomMotion = ((PhaseStabilize and (SSA1Low_ or SIMStatusNotOk_)) or CSCStatusFault_ or CDAYMagMomMax_ or CDAZMagMomWrongSign_)`
	Fault detector:
4	`DetCDAZStatusFault = (CDAZon and CDAEnabledon and (CDAZCurrentMeasWrong_ or ((CDAZMagMomWrongSign_ or CDAZMagMomZero_) and PhaseDetumbling)))`
	Decision Logic:
5	`CDAZOn = (not DetCDAZStatusFault)`

(rule 1) has only an effect on the system if the two reconfiguration switches CDAZ and CDAEnabled are both On (rule 2). In that case, it causes a random motion of the satellite along with some other faults that have the same end-effect (rule 3). The coil driver sign fault is detectable, but only in the detumbling phase and if CDAZ is On (rule 4). The decision logic reconfigures by disabling CDAZ switch (rule 5).

The analysis works, as explained on page 39, on the fact that the propagated fault (CDAZMagMomWrongSign_) will be *bound to false* by the rules. This is clear because if it was true then CDAZOn will be false by rule 5, and this will lead to CDAZMagMomWrongSign_=False by rule 2. Based on this observation, the following is analyzed:

Fault analysis. The rule base is searched for bindings on the propagated fault variables. The list of variables bound to false indicate the faults that are successfully reconfigured. Table 5.13 shows the results of the analysis, performed separately for detumbling and stabilization. It is clear from the table that all faults are accommodated in

Table 5.13: *Analysis of the completeness of the Ørsted ACS decision logic. The table shows the component faults that will be handled (OK) and those that will not be handled (Not OK).*

Fault	Detumbling	Stabilization
CSCStatusFault	OK	OK
SIMStatusNotOk	–	OK
SS<i>Low	–	OK
SS<i>High	–	OK
SST<i>StatusFault	–	OK
CD<s><i>ShuntShort	OK	OK
CD<s><i>ShuntDisc	OK	OK
CD<s><i>T5Short	OK	OK
CD<s><i>T5Open	OK	OK
CD<s><i>SignFailedPos	OK	Not OK
CD<s><i>SignFailedNeg	OK	Not OK
CD<s><i>CoilShort	OK	Not OK
CD<s><i>CoilDisc	OK	OK
CD<s><i>PosTermShortGND	OK	OK
CD<s><i>NegTermShortGND	OK	OK
CD<s><i>T1Short	OK	Not OK
CD<s><i>T2Short	OK	Not OK
CD<s><i>T1Open	OK	OK
CD<s><i>T2Open	OK	OK

detumbling, but some are not handled in the stabilization phase. The reason for this is that these faults cannot be detected as explained earlier in section 5.4.1.

End-effect analysis. An inspection of the rule-base for possible end-effects yields the results for the two operational phases shown in table 5.14. The table shows that the three

Table 5.14: *Analysis of the completeness of the Ørsted ACS decision logic. The table shows which end-effects are prevented from occurring with the implemented decision logic and which can still happen.*

End-effect	Detumbling	Stabilization
RandomMotion	Impossible	Possible
LargeError	Impossible	Possible
SmallError	Impossible	Possible
PowerHigh	Impossible	Impossible

end-effects RandomMotion, LargeError, and SmallError can still happen.

A further search for faults that cause these possible end-effects during the stabilization phase gives the fault distribution shown in table 5.15. The faults that cause

Table 5.15: *Analysis of the faults that can cause the possible end-effects during the stabilization phase shown in table 5.14.*

RandomMotion	LargeError	SmallError
CDAZSignFailedPos	CDAYCoilShort	CDAXCoilShort
CDAZSignFailedNeg	CDAYT1Short	CDAXSignFailedPos
	CDAYT2Short	CDAXSignFailedNeg
		CDAXT1Short
		CDAXT2Short
		CDAYSignFailedPos
		CDAYSignFailedNeg
		CDAZCoilShort
		CDAZT1Short
		CDAZT2Short

RandomMotion and LargeError are required to be handled (see section 5.3.3), but as explained in section 5.4.1, this is not feasible for the Ørsted satellite. This means that the requirements are relaxed on this point.

5.7 Supervisor Implementation in Software

The Ørsted ACS, including the FTCS supervisor, is implemented in the object oriented multi-tasking language Ada. Sophisticated programming techniques are not employed, so the experience gained from the realization can be useful as a general assessment on problems and solutions encountered in such systems. This section presents some issues relevant to the implementation of the supervisor task in the three layer structure presented in section 3.1.

5.7.1 The Supervisor Task

The ACS supervisor is assigned a separate task and runs independently of the control level task. The sampling time is 20 seconds which has been derived from the *reconfiguration requirements* in section 5.3.4. The following pseudo-code explains the supervisor loop:

```
Supervisor_task
 1  Infinite loop
 2     Receive synchronuous command
 3     If parameter modification command (CmdModify) then
 4        Stop control level task
 5        Modify parameters
 6        Restart control level task
 7     Call SEM inference with commands
 8     Update control level with new states (including start and stop)
 9     Process SEM outputs
10     Send command verification to ground
11     If reset command (CmdReset) then call initialization subroutines
12  Every 20th second:
13     Call detector subroutines
14     Decrement timer and collect timeout signals (TimSIMTimeout)
15     Override detector outputs with mask from ground
16     Call SEM inference with masked detector outputs
17     Update control level with new states
18     Process SEM outputs
19     Collect and send housekeeping every minute, if enabled
20  end loop
```

The main loop executes the detectors periodically (line 12) and receives commands from the on-board data handler through a synchronous call (line 2). The SEM inference of detector outputs and commands is thus separated into two different subroutines (lines 7 and 16), both operating on the same SEM states. The additional supervisor tasks are reconfiguration of the control level by transferring the new SEM states (lines 8 and 17), execution handling (start, stop) of the periodic control level and supervisor tasks (lines 4, 6 and 8), handling of parameter updates in ACS (lines 3-6), and transmitting information to ground (housekeeping, command verification, and state updates in lines 10 and 19).

The list of general requirements on page 109 includes a mechanism for disabling and enabling the on-board autonomous functions. This is implemented with a detection *mask* that makes it is possible to override the boolean values of the detector events before the vector is presented for the inference (line 15). The detector mask is changed by a parameter update command from ground. It provides full control over the SEMs from ground.

It has not been feasible to implement the decision logic inference with automatically generated code from VisualState™ due to memory limitations. Instead, the above methodology with design and verification of the SEMs using Beologic™ AIT and VisualState™ has been followed, while the implemented code was developed as an Ada counterpart that was verified with a test sequence developed in Beologic™.

5.7.2 Synchronization between the Control Level and the Supervisor Tasks

The control and data flows between the supervisor and control level tasks are presented in figure 5.13. The supervisor cycle with FDIR can be interrupted by a synchronous

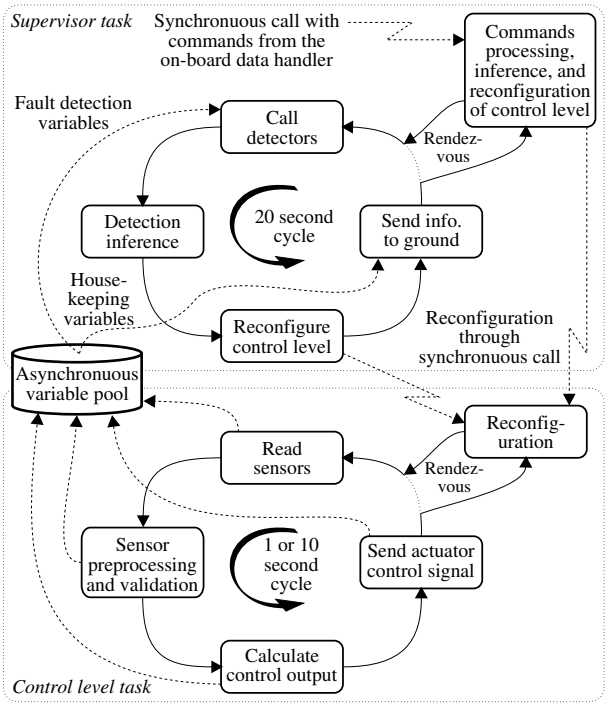


Figure 5.13: *Control and data flow diagram between the supervisor task and the control level task. Dashed lines indicate data flow and solid lines indicate control flow.*

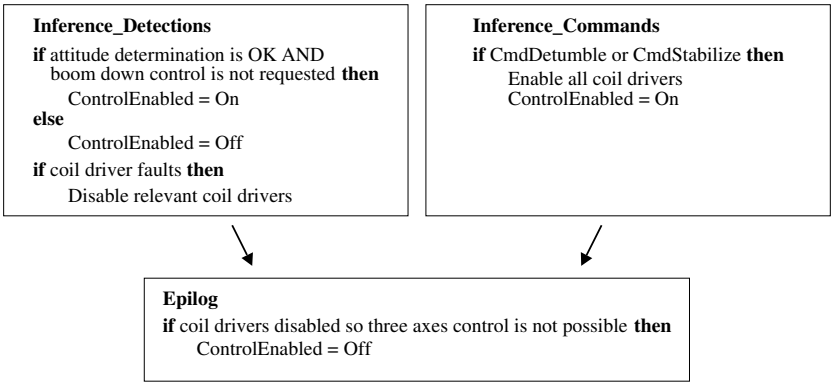
rendezvous with commands from the on-board data handler. The control level task runs a one second (detumbling) or 10 seconds (stabilization) cycle that can be interrupted by a synchronous rendezvous with reconfiguration commands from the supervisor task.

The variables needed for fault detection and ground information are transferred asynchronously from the control level task to the supervisor task through a *variable pool* that is protected by a semaphore. The semaphore ensures synchronism between more variables when this is required. Also the results of the sensor validity checks performed by the control level (see section 5.4.1) are stored in the variable pool.

5.7.3 Sequential versus Parallel SEM Inference

The SEM inference mechanism is implemented in traditional sequential code using for-loops and if-then structures. It would have been advantageous to realize it with a number of matrices containing the SEM transition rules, and use a common state transition subroutine to perform the inference. This solution was rejected because the Ada compiler creates an instantiation for each matrix and this causes the entire SEM code to swell up to all-most 5 times the size of the sequential counterpart (7 kbyte against 1.4 kbyte).

The disadvantage with sequential code is that internal dependencies between SEMs are difficult to overview. The ControlEnabled SEM serves as an example on this. This SEM is used by both the detection inference and the command inference subroutines as indicated with the following pseudo code:



Both subroutines call the Epilog in the bottom box. If the coil drivers are disabled by Inference_Detections, so three axes control is not possible, then ControlEnabled is turned Off in the Epilog. The coil drivers are re-enabled by ground command (CmdDetumble or CmdStabilize). It is necessary that Inference_Commands also turns ControlEnabled On because the Epilog is only able to turn it Off. If this was neglected, the controller will not be enabled before the next supervisor cycle. This illustrates the undesirable considerations required for sequential implementations.

5.8 Test of the Fault Tolerant Control System

The fault tolerant control system for the Ørsted ACS was designed with the methodology presented in this thesis, but this does not replace a well structured test. This section discusses how the supervisor algorithms can be tested on different test levels and also how the supervisor can be designed to improve testability of the ACS subsystems.

The ACS has been tested with the ordinary sequence of *module test, integration test* and *acceptance test*. The module test involves only the software subsystems. The integration test involves the entire on-board software (data handling and ACS) and elec-

trical interface to sensors, actuators, and computers. The acceptance test involves all ACS relevant instruments on the satellite.

5.8.1 Fault Detector Tests

A Matlab™ simulation program of the satellite dynamics which includes fault models provides inputs for the FDI algorithm *module* tests. The tests are executed with dedicated test drivers and test stubs that are linked to the module under test at compile-time.

The *integration* test is also based on simulated sensor values. A test bench that emulates sensors and actuators through the electrical interfaces serves as a platform for a hardware-in-the-loop test, where the Matlab™ simulated sensor values are presented in real-time by a PC. The Matlab™ simulation is executed off-line so the test is not a closed loop test. These tests ensure correct synchronization between the control level task and the supervisor task as well as consistency of the interfaces. It is possible to imitate all the considered failure modes using this equipment.

The final *acceptance* test is more involved because it is very difficult and expensive to create a realistic environment around the satellite. It is furthermore prohibited to induce deliberate faults in the equipment. This makes it literally impossible to test detection of faults, but it is likewise important to test absence of false alarms. In the cases where a suitable close-to-normal condition can be established, tests for incorrect intervention from the supervisor can be performed. The detumbling phase can, for example, be tested in this way. Only the magnetic field measurement and the magnetorquers are active in this period, so the test covers only the CSC and CD/MTQ FDI algorithms. The test is performed simply by activating the detumbling control and then rotate the satellite slowly. The only non-space phenomena in this setup is a stronger magnetic field on ground than in space and a 50 Hz fluctuation of the field caused by nearby power lines.

5.8.2 Supervisor Tests

The *module* test of the decision logic's SEMs is based on a sequence of input events that has been developed during the SEM design. The inference algorithm is linked to a test driver that generates inputs for the test sequence and also a list of desired outputs and state transitions. The performance of the SEM inference is then compared to the desired functionality.

Correct operation of the supervisor loop, presented in section 5.7.1, is verified in the *integration* test using the hardware-in-the-loop test bench. The most important issue here is the synchronization between the three tasks involved (control level, supervisor, and the on-board data handler's command interface), where possible multi-tasking deadlocks can terminate the tasks. The integration test is also used to verify that the fault reconfiguration is appropriate and performed in time.

The only supervisor aspects, that can be tested during the *acceptance* test, are the commands/monitoring treatment and the ability to reconfigure the control level. The

latter is achieved by forcing desired state transitions with the detector *mask* (see section 5.7.1) and observe correct reconfiguration of the control level in different situations.

5.8.3 ACS Performance Tests

The supervisor is an integrated part of the ACS, and it is very important to design the system, so the supervisor supports testability of the control level. A control level test typically requires some subsystems to be forced into a specific mode whereas others shall be reconfigurable by the supervisor. An example on this is the test of the secondary attitude determination mode (refer to figure 5.6 on page 119). The AttDetMode switch must be forced to "Sec" during the test, but the SVectorEnabled switch shall still be operable by the supervisor. The SVectorEnabled switch enables and disables the sun vector input depending on the eclipse condition, and it is a part of the test to include these transitions. Such requirements are supported by the present supervisor design, where the detector *mask* is able to disable selected detectors and thereby force desired settings of the control level.

The entire ACS is designed to be very flexible with respect to testing, debugging, and maintenance. All internal parameters, both in the control level and the detectors, can be modified by ground command and most variables can be diagnosed. This is used in a set of formalized test sequences, implemented and executed from the ground station. Each test sequence is dedicated to perform a specific subsystem test. Built-in test drivers have been considered, but discarded due to memory limitations.

5.8.4 Test Results

As an illustration on the test of the Ørsted satellite's FTCS, a single test result is presented where a particular fault is detected, isolated, and accommodated. Several tests have been performed with a Matlab™ simulation of the satellite dynamics, and the real on-board software have been verified in using the open-loop hardware-in-the-loop test setup.

The fundamental purpose of testing is to verify fulfillment of the *reconfiguration requirement* stated in section 5.3.4. This includes an assessment on the time to detect and reconfigure and the amount of performance degradation the fault caused before the system was reconfigured. Also situations with potential false alarms and missed detections are investigated.

The test presented here concerns the detumbling phase that has the purpose to damp the initial rotational energy of the tumbling satellite and align the satellite's z-axis with the geomagnetic field vector. The ZB coil driver fails after 20 minutes so only positive currents are producible (CDZBSignFailedPos). This fault is detected and the coil system is reconfigured so the ZB coil is disabled and the ZA coil is used instead. Figure 5.14 shows the two situations where the fault is handled and where it is not handled. It is clear from the figure that the satellite will continue tumbling with approximately 20 times the nominal angular velocity if the fault is not accommodated. This end-effect

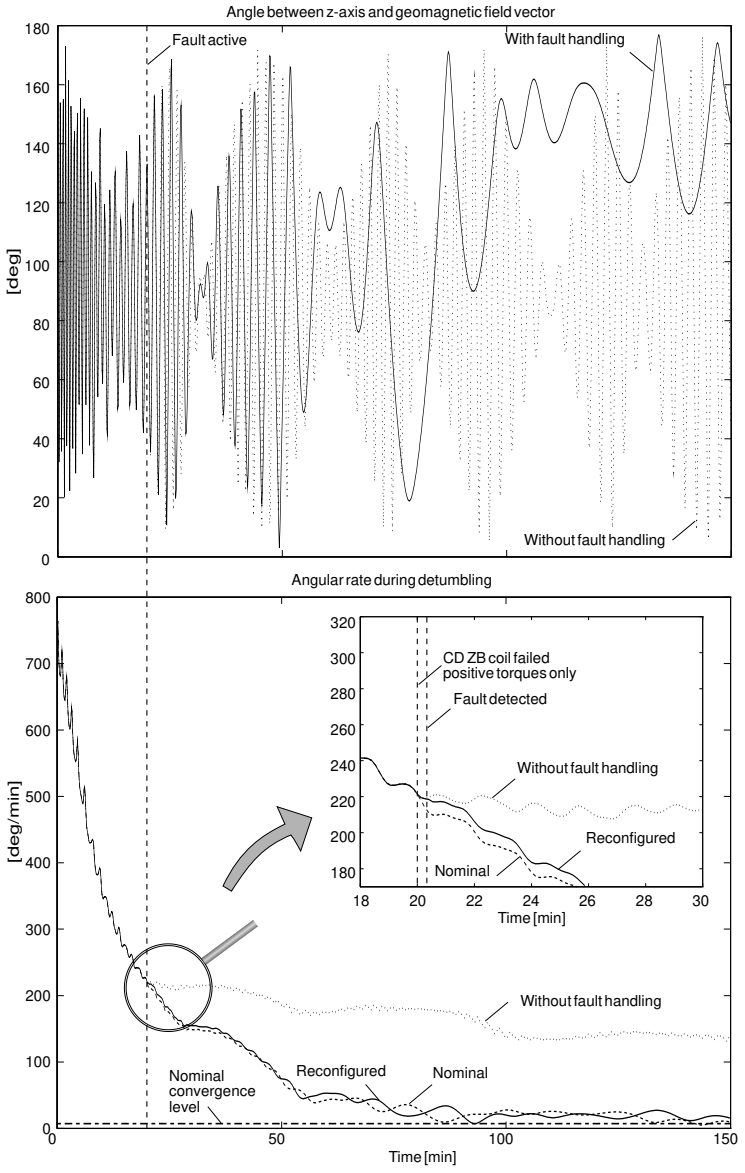


Figure 5.14: *Example on fault handling on the Ørsted satellite in the detumbling phase. The top graph shows the angle between the satellite and the local geomagnetic field vector, which is supposed to approach 180 deg. The bottom graph shows the amplitude of the angular velocity. When the fault is not accommodated, the satellite will continue tumbling and thereby prevent boom deployment.*

will prevent the boom from deploying successfully. When the fault is accommodated the performance is maintained on the same level as if no fault would have happened.

The detection time is 60 seconds, which is fast enough to ensure that the angular rate is changed no more than 7 deg/min, which is far below the requirement of 80 deg/min stated on page 119.

5.9 Summary

This chapter described the development of fault tolerance in the Ørsted satellite attitude control system based on the general procedure introduced in chapter 3. The Ørsted satellite is realized with a low level of redundancy, so the methods typically used for large and expensive satellites are not applicable in this case. This chapter illustrated how system availability and performance can be improved by considering faults in a systematic way.

It was shown that a simple logic modelling of the fault effect propagation through subsystems is feasible when the system is not too complex. If it is not possible to determine the end-effects from faults by experience or simple analysis, as is the case for the satellite dynamic motion, then it is necessary to apply numerical simulation. It is up to the design engineer whether this step is required or not, but often such simulation models are available from the controller design.

The major difference between the Ørsted application and the benchmark case presented in section 3.3 lies in the complexity of the supervisor. This chapter illustrated how a complex decision logic can be designed as a set of state-event machines (SEMs) and verified on three levels: Consistency of the decision logic's SEM rules, consistency between the decision logic's SEM rules and a SEM description of the control system, and finally completeness of the fault handling. Implementation in a multi-tasking system were presented with a discussion on the separation of the FTCS elements into several real-time tasks. It was furthermore realized, that a systematic implementation of the supervisor was not feasible due to memory limitations, so the final code had to be implemented as the more compact if-then sentences. This problem will, probably, vanish with the progress of larger computers that, with some years lag, become available in space qualified versions.

Finally, it was discussed how the fault handling system can be tested on various test levels, and specifically how the FTCS can be designed to improve testability.

Chapter 6

Conclusion and Recommendations

This thesis considered a number of aspects for the design and implementation of fault tolerant control systems. General guidelines were given to each step in the development process with the purpose to make the approach applicable for real-life systems. As a general conclusion, a number of innovative ideas have been developed from case studies, but they need to be matured and proved for industrial applicability by further realistic examples and application studies. This chapter summarizes the major achievements of the work and suggests directions for future investigations.

6.1 Conclusion

The following conclusions are drawn on the accomplishments of this thesis:

- A general development methodology for FTCS analysis and design was presented in chapter 3. An eight-step procedure was introduced with reference to the standard water-fall model, where each step covers an issue in the FTCS development. A systematic analysis methodology was presented that leads to consistent specifications for the design of algorithms for fault detection, fault accommodation, and supervisory control. It was shown that a systematic analysis of potential failures, carried out in the preliminary analysis phase, and a *logic modelling* of the fault effect propagation through subsystems can be used as a valuable assessment on fault aspects. This preliminary analysis draws the attention to possible severe fault effects and the necessities/possibilities for implementing fault tolerance in the system. The case studies in section 3.3 and chapter 5 illustrated that the logic fault propagation models implemented in a software tool-kit establish a practical instrument for analysis of the relationship between component faults and their

end-effects. It also supports a preliminary assessment on the detectability and isolability of faults with a particular detection scheme (section 3.3.7). When the fault propagation models are combined with the design of decision logic for fault handling, this framework provides a powerful tool for checking *completeness* of the decision logic with respect to fault coverage. This completeness check iss only reasonable if the underlying fault analysis is sufficient and the logic models are adequate and correct. The suggested development methodology will therefore be applicable for industrial use, when a library containing component failure modes and fault propagation models has been developed.

- The method using logic models for fault propagation was shown to provide a convenient tool for analyzing the behaviour of *feedback loops*. It was shown that this analysis can be used to determine if a fault causes the serious effect of an oscillation in the physical system. Feedback loops in the logic models lead to cyclic graphs, and these have been recognized by graph theorists as a problematic issue. A simple and sufficient solution for identifying such loops in connection with fault propagation analysis was given in section 3.2.2. This method was proven successful in section 3.3.3 for a simple example with two cascaded controllers, but full applicability for large complex systems with multiple control loops must be further investigated.

- Implementation of the fault handling decision logic as *state-event machines* (SEMs) was shown in section 5.5 to be adequate for fault handling and on the same time suitable to manage existing requirements for operational control. The thesis has not considered automatic translation of the specifications for the decision logic into SEMs, but selected design guidelines were given on typical problems. Section 5.6 showed the advantages of applying an established method for realization (such as SEMs), because a number of potential design and implementation errors can be automatically identified by existing software tools. The methods described in the thesis can be used in connection with automatic generation of executable code, which is vital to ensure agreement between the requirement specification and the final implementation.

- A three-level architecture for the implementation of an FTCS was applied to the benchmark case in section 3.4 and to the micro satellite in chapter 5. The three levels are the bottom layer with reconfigurable control, the second layer with fault detectors, and the third layer with decision logic. This arrangement supports modular implementation and test and has proven to be especially fruitful for the satellite control system development, where individual programmers have implemented and tested each layer.

- A comparison between nine different methods for detection of two realistic faults in the electro-mechanical benchmark equipment was provided in chapter 4. This study demonstrated fundamental equivalence between several approaches and full

potential for detection of simple faults that can be described satisfactorily as additive inputs to a linear model. It is more problematic when standard techniques are used to solve realistic problems that involve dominating nonlinear characteristics like saturation, and where it is not sufficient to assume the fault to be an unknown additive input. The nine research groups that contributed to the benchmark test have shown remarkable creativity to either modify the standard techniques to accommodate this problem or design dedicated schemes that accept structural information. The thesis contributed specifically with simple and sufficient algorithms for detection of the considered faults, where standard engineering knowledge was used to design an appropriate detection scheme that uses only the necessary process measurements with corresponding models of the system characteristics.

- A complete design and implementation of an FTCS for the Ørsted satellite in chapter 5 illustrated the potentials of the development methodology introduced in this thesis. It was shown how the logic fault propagation analysis of a complex system can be broken up into subsystems and be combined later with the decision logic for a completeness check as mentioned above. The satellite example also showed the limitations of the logic modelling framework. If it is not possible to predict the effect of a fault in a complicated dynamic system, as is the case for the satellite, it may be necessary to apply numerical simulation or advanced analysis techniques. Another problem experienced in the satellite application was that a systematic implementation of the decision logic SEMs is infeasible due to memory consumption. Both automatically generated code and a systematic matrix implementation required substantially more memory than ordinary if-then structures. This is a potential degradation of the supervisor reliability due to the risk of implementation errors and also unanticipated dependencies in the sequential if-then structure. In spite of these hurdles, it showed up to be advantageous to organize the entire FTCS with the decision logic as a central function, because late modifications in the requirements for the supervisor could easily be implemented and verified.

6.2 Recommendations

It was the purpose with this thesis to give a coherent set of practical guidelines that support the implementation of fault tolerance in control systems. For further extensions and refinements of the methodology, the following recommendations are given:

- The logic fault propagation modelling framework introduced in section 3.2.2 does not include system dynamics. It is possible to include a coarse description of the *temporal behaviour* of fault propagation by including a minimum and maximum time in the description. This could be used to extend the completeness check to also guarantee that the fault handling system will reconfigure the process within the temporal requirements. Misra (1994) uses this approach in connection with

diagnosis of dynamical systems. The software tool-boxes used in this thesis do not support this analysis.

- This thesis has not considered formal methods for the design of the supervisor's decision logic. Supervisory control theory (SCT), used in the field of discrete event systems (DES), is an interesting research area, where formal methods provide automatic synthesis of supervisors and completeness checks (see section 2.3.4 for a discussion). While SCT may be unapplicable for practical use, the same fundamental principles are used in procedural control theory (PCT), so combined with a suitable tool (e.g. Grafcet) this framework may be suitable for a systematic design of the decision logic in FTCS.

- It has been sufficient for the case studies in this thesis to base the fault handling decisions solely on *boolean* fault events, i.e. whether a fault is detected present or not. Additional information like time of occurrence and magnitude of the fault effect may also be used if required. It can, furthermore, be advantageous to allow non-boolean information as input to the supervisor to indicate the degree of membership to true or false. Such information could be derived from the significance of the fault and the credibility of the detection. *Fuzzy* techniques could then be applied for the design of the supervisor decision logic.

- The software tool-boxes used in this thesis to support development of FTCSs are not designed for the purpose, so a suitable software environment must be established before the methodology becomes industrially applicable. Such a tool-box should be based on a library containing failure modes of standard components and corresponding fault propagation models. In this software, it shall be possible to connect the component models into a complete system description, identify and handle feedback loops, analyse fault-effect relationships, synthesize decision logic for fault handling and operational control, make completeness checks of the decision logic, generate test sequences, generate executable code, and make systematic documentation.

Appendix A

Benchmark Beologic™ Rules

This appendix contains listings of the Beologic™ Array Inference Toolbox (AIT) rules for the benchmark application.

A.1 Fault Propagation Graph - without "Oscillation"-Discrepancy

```
PosMeasLow          =   ( (not (VnegWireDisc or VoutWireDisc) and PosLow) or
                        VposWireDisc )
PosMeasConst        =   ( (not (VposWireDisc or VnegWireDisc) and PosConst) or
                        VoutWireDisc )
PosMeasHigh         =   ( (not (VposWireDisc or VoutWireDisc) and PosHigh) or
                        VnegWireDisc )
VelRefDesLow        =   ((PosRefLow and not PosCtrlLoopFault) or
                        (PosCtrlLoopFault and (PosMeasHigh or (PosMeasConst and
                        PosRefLow) )))
VelRefDesZero       =   (PosRefConst and not PosCtrlLoopFault)
VelRefDesHigh       =   ((PosRefHigh and not PosCtrlLoopFault) or
                        (PosCtrlLoopFault and (PosMeasLow or (PosMeasConst and
                        PosRefHigh) )))
VelRefLow           =   (not VelRefWireDisc and VelRefDesLow)
VelRefZero          =   (VelRefWireDisc or (not VelRefWireDisc and
                        VelRefDesZero))
VelRefHigh          =   (not VelRefWireDisc and VelRefDesHigh)
VelMeasLow          =   ((not TachoWireDisc and VelLow))
VelMeasZero         =   (TachoWireDisc or VelZero)
VelMeasHigh         =   ((not TachoWireDisc and VelHigh))
CurRefLow           =   ((VelRefLow and not VelCtrlLoopFault) or
                        (VelCtrlLoopFault and (VelMeasHigh or (VelMeasZero and
                        VelRefLow) )))
CurRefZero          =   (VelRefZero and not VelCtrlLoopFault)
CurRefHigh          =   ((VelRefHigh and not VelCtrlLoopFault) or
                        (VelCtrlLoopFault and (VelMeasLow or (VelMeasZero and
                        VelRefHigh) )))
PowerFailed         =   ((EndSwitchNegDisc and CurRefLow) or (EndSwitchPosDisc
                        and CurRefHigh)),
                        "PowerFailed is an intermediate variable"
```

```
(not CurZero)      ->   not ( EndSwitchNegDisc or EndSwitchPosDisc ),
                        "This rule ensures that EndSwitchNegDisc and
                        EndSwitchPosDisc are both false if CurZero is false.
CurLow             =    ((not PowerFailed and CurRefLow))
CurZero            =    ((not PowerFailed and CurRefZero) or PowerFailed)
CurHigh            =    ((not PowerFailed and CurRefHigh))
VelLow             =    (not BrakeFailedON and CurLow)
VelZero            =    (CurZero or BrakeFailedON)
VelHigh            =    (not BrakeFailedON and CurHigh)
PosLow             =    VelLow
PosConst           =    VelZero
PosHigh            =    VelHigh
VelCtrlLoopFault   =    (TachoWireDisc or BrakeFailedON or EndSwitchPosDisc or
                        EndSwitchNegDisc ),
                        "Manual assignment of faults that appear inside the
                        velocity control loop"
PosCtrlLoopFault   =    ( VelCtrlLoopFault or VposWireDisc or VnegWireDisc or
                        VoutWireDisc or VelRefWireDisc ),
                        "Manual assignment of faults that appear inside the
                        position control loop"
oneof(PosLow PosHigh PosConst)
oneof(PosRefLow PosRefHigh PosRefConst)
oneof(PosMeasLow PosMeasHigh PosMeasConst)
oneof(VelMeasLow VelMeasHigh VelMeasZero)
oneof(VelRefDesLow VelRefDesHigh VelRefDesZero)
oneof(VelRefLow VelRefHigh VelRefZero)
oneof(CurRefLow CurRefHigh CurRefZero)
oneof(CurLow CurHigh CurZero)
oneof(VelLow VelHigh VelZero)
oneof (VelRefWireDisc TachoWireDisc VposWireDisc VnegWireDisc VoutWireDisc
BrakeFailedON EndSwitchPosDisc EndSwitchNegDisc),
                        "Limit to single fault analysis.  One and only fault is
                        allowed true."
```

A.2 Fault Propagation Graph Rules - with “Oscillation”-Discrepancy

The FPG above is expanded with an “oscillation” discrepancy. Changes are emphasized in boldface.

```
PosMeasLow         =    ( (not (VnegWireDisc or VoutWireDisc) and PosLow) or
                        VposWireDisc )
PosMeasConst       =    ( (not (VposWireDisc or VnegWireDisc) and PosConst) or
                        VoutWireDisc )
PosMeasHigh        =    ( (not (VposWireDisc or VoutWireDisc) and PosHigh) or
                        VnegWireDisc )
PosMeasOscil       =    (not (VposWireDisc or VnegWireDisc or VoutWireDisc) and
                        PosOscil)
VelRefDesLow       =    ((PosRefLow and not PosCtrlLoopFault) or
                        (PosCtrlLoopFault and (PosMeasHigh or (PosMeasConst and
                        PosRefLow) )))
VelRefDesZero      =    (PosRefConst and not PosCtrlLoopFault)
VelRefDesHigh      =    ((PosRefHigh and not PosCtrlLoopFault) or
                        (PosCtrlLoopFault and (PosMeasLow or (PosMeasConst and
                        PosRefHigh) )))
VelRefDesOscil     =    (PosCtrlLoopFault and PosMeasOscil)
VelRefLow          =    (not VelRefWireDisc and VelRefDesLow)
VelRefZero         =    (VelRefWireDisc or (not VelRefWireDisc and
                        VelRefDesZero))
```

```
VelRefHigh          =    (not VelRefWireDisc and VelRefDesHigh)
VelRefOscil         =    (not VelRefWireDisc and VelRefDesOscil)
VelMeasLow          =    ((not TachoWireDisc and VelLow))
VelMeasZero         =    ((not TachoWireDisc and VelZero))
VelMeasHigh         =    ((not TachoWireDisc and VelHigh))
VelMeasOscil        =    TachoWireDisc or VelOscil
CurRefLow           =    ((VelRefLow and not VelCtrlLoopFault) or
                         (VelCtrlLoopFault and (VelMeasHigh or (VelMeasZero and
                         VelRefLow) )))
CurRefZero          =    (VelRefZero and not VelCtrlLoopFault)
CurRefHigh          =    ((VelRefHigh and not VelCtrlLoopFault) or
                         (VelCtrlLoopFault and (VelMeasLow or (VelMeasZero and
                         VelRefHigh) )))
CurRefOscil         =    ((VelRefOscil and not VelCtrlLoopFault) or
                         (VelCtrlLoopFault and VelMeasOscil))
PowerFailed         =    ((EndSwitchNegDisc and CurRefLow) or (EndSwitchPosDisc
                         and CurRefHigh)),
                         “PowerFailed is an intermediate variable”
(not CurZero)       ->   not ( EndSwitchNegDisc or EndSwitchPosDisc ),
                         "This rule ensures that EndSwitchNegDisc and
                         EndSwitchPosDisc are both false if CurZero is false.
CurLow              =    ((not PowerFailed and CurRefLow))
CurZero             =    ((not PowerFailed and CurRefZero) or PowerFailed)
CurHigh             =    ((not PowerFailed and CurRefHigh))
CurOscil            =     ((not PowerFailed and CurRefOscil))
VelLow              =    (not BrakeFailedON and CurLow)
VelZero             =    (CurZero or BrakeFailedON)
VelHigh             =    (not BrakeFailedON and CurHigh)
VelOscil            =    (not BrakeFailedON and VelMeasOscil)
PosLow              =    VelLow
PosConst            =    VelZero
PosHigh             =    VelHigh
PosOscil            =    VelOscil
VelCtrlLoopFault    =    (TachoWireDisc or BrakeFailedON or EndSwitchPosDisc or
                         EndSwitchNegDisc ),
                         "Manual assignment of faults that appear inside the
                         velocity control loop"
PosCtrlLoopFault    =    ( VelCtrlLoopFault or VposWireDisc or VnegWireDisc or
                         VoutWireDisc or VelRefWireDisc ),
                         "Manual assignment of faults that appear inside the
                         position control loop"
oneof(PosLow PosHigh PosConst PosOscil)
oneof(PosRefLow PosRefHigh PosRefConst)
oneof(PosMeasLow PosMeasHigh PosMeasConst PosMeasOscil)
oneof(VelMeasLow VelMeasHigh VelMeasZero VelMeasOscil)
oneof(VelRefDesLow VelRefDesHigh VelRefDesZero VelRefDesOscil)
oneof(VelRefLow VelRefHigh VelRefZero VelRefOscil)
oneof(CurRefLow CurRefHigh CurRefZero CurRefOscil)
oneof(CurLow CurHigh CurZero CurOscil)
oneof(VelLow VelHigh VelZero VelOscil)
oneof (VelRefWireDisc TachoWireDisc VposWireDisc VnegWireDisc VoutWireDisc
BrakeFailedON EndSwitchPosDisc EndSwitchNegDisc),
                         "Limit to single fault analysis.  One and only fault is
                         allowed true."
```

A.3 Fault Detection and Isolation Rules

```
DetVelPos          =   ((PosMeasLow and VelMeasZero) or (PosMeasLow and
                       VelMeasHigh) or (PosMeasConst and VelMeasLow) or
                       (PosMeasConst and VelMeasHigh) or (PosMeasHigh and
                       VelMeasLow) or (PosMeasHigh and VelMeasZero) or
                       PosOscil),
                       "Detector:  Detect deviations between velocity and
                       position measurements bas ed on model of gear and
                       integrator"
DetVelRefVel       =   ((VelRefDesLow and VelMeasZero) or (VelRefDesLow and
                       VelMeasHigh) or (VelRefDesZero and VelMeasLow) or
                       (VelRefDesZero and VelMeasHigh) or (VelRefDesHigh
                       and VelMeasLow) or (VelRefDesHigh and VelMeasZero) or
                       VelOscil),
                       "Detector:  Detect deviations between velocity reference
                       and velocity measur ement based on model of motor"
DetVelCur          =   ((CurLow and VelMeasZero) or (CurLow and VelMeasHigh) or
                       (CurZero and VelMeasLow) or (CurZero and VelMeasHigh) or
                       (CurHigh and VelMeasLow) or (CurHigh and VelMeasZero)),
                       "Detector:  Detect deviations between velocity
                       measurement and current measu rement based on model of
                       motor"
DetVelRefCur       =   ((CurLow and VelRefDesZero) or (CurLow and VelRefDesHigh)
                       or (CurZero and VelRefDesLow) or (CurZero and
                       VelRefDesHigh) or (CurHigh and VelRefDesLow) or (CurHigh
                       and VelRefDesZero) or CurOscil),
                       "Detector:  Detect deviations between velocity reference
                       and current based o n model of motor"
IsoPosMeasFault    =   (DetVelPos and not DetVelRefVel),
                       "Isolation:  Isolate position measurement fault"
IsoVelCtrlFault    =   (DetVelPos and DetVelRefVel),
                       "Isolation:  Isolate velocity measurement fault"
IsoOtherFault      =   (DetVelRefVel and not DetVelPos),
                       "Isolation:  Isolate power drive, end-switch, or motor
                       faults"
```

A.4 Complete Rule Base with FPG, FDI, Decision Logic, and Reconfiguration

The complete logic model includes switches for reconfiguration. The extensions and changes to the FPG in A.2 are emphasized in boldface.

Fault propagation graph:

```
PosMeasLow         =    ( (not (VnegWireDisc or VoutWireDisc) and PosLow) or (
                       VposWireDisc and not PosEstim) )
PosMeasConst       =    ( (not (VposWireDisc or VnegWireDisc) and PosConst) or (
                       VoutWireDisc and not PosEstim) )
PosMeasHigh        =    ( (not (VposWireDisc or VoutWireDisc) and PosHigh) or
                       (VnegWireDisc and not PosEstim) )
PosMeasOscil       =    (not (VposWireDisc or VnegWireDisc or VoutWireDisc) and
                       PosOscil)
VelRefDesLow       =    ((PosRefLow and not PosCtrlLoopFault) or
                       (PosCtrlLoopFault and (PosMeasHigh or (PosMeasConst and
                       PosRefLow) )))
VelRefDesZero      =    (PosRefConst and not PosCtrlLoopFault)
```

```
VelRefDesHigh       =    ((PosRefHigh and not PosCtrlLoopFault) or
                         (PosCtrlLoopFault and (PosMeasLow or (PosMeasConst and
                         PosRefHigh) )))
VelRefDesOscil      =    (PosCtrlLoopFault and PosMeasOscil)
VelRefLow           =    (not VelRefWireDisc and VelRefDesLow and not
                         VelCtrlBypass)
VelRefZero          =    (VelRefWireDisc or (not VelRefWireDisc and VelRefDesZero)
                         and not VelCtrlBypass)
VelRefHigh          =    (not VelRefWireDisc and VelRefDesHigh and not
                         VelCtrlBypass)
VelRefOscil         =    (not VelRefWireDisc and VelRefDesOscil and not
                         VelCtrlBypass)
(VelCtrlBypass and VelRefDesLow) -> CurRefLow
(VelCtrlBypass and VelRefDesZero)-> CurRefZero
(VelCtrlBypass and VelRefDesHigh)-> CurRefHigh
(VelCtrlBypass and VelRefDesOscil) -> CurRefOscil
VelMeasLow          =    ((not TachoWireDisc and VelLow))
VelMeasZero         =    ((not TachoWireDisc and VelZero))
VelMeasHigh         =    ((not TachoWireDisc and VelHigh))
VelMeasOscil        =    TachoWireDisc or VelOscil
((VelRefLow and not VelCtrlLoopFault) or (VelCtrlLoopFault and (VelMeasHigh or
(VelMeasZero and VelRefLow) ))) -> CurRefLow
(VelRefZero and not VelCtrlLoopFault) -> CurRefZero
((VelRefHigh and not VelCtrlLoopFault) or (VelCtrlLoopFault and (VelMeasLow or
(VelMeasZero and VelRefHigh) ))) -> CurRefHigh
((VelRefOscil and not VelCtrlLoopFault) or (VelCtrlLoopFault and VelMeasOscil)) ->
CurRefOscil
PowerFailed         =    ((EndSwitchNegDisc and CurRefLow) or (EndSwitchPosDisc
                         and CurRefHigh)),
                         "PowerFailed is an intermediate variable"
(not CurZero)       ->   not ( EndSwitchNegDisc or EndSwitchPosDisc ),
                         "This rule ensures that EndSwitchNegDisc and
                         EndSwitchPosDisc are both false if CurZero is false.
CurLow              =    ((not PowerFailed and CurRefLow))
CurZero             =    ((not PowerFailed and CurRefZero) or PowerFailed)
CurHigh             =    ((not PowerFailed and CurRefHigh))
CurOscil            =    ((not PowerFailed and CurRefOscil))
VelLow              =    (not BrakeFailedON and CurLow)
VelZero             =    (CurZero or BrakeFailedON)
VelHigh             =    (not BrakeFailedON and CurHigh)
VelOscil            =    (not BrakeFailedON and VelMeasOscil)
PosLow              =    VelLow
PosConst            =    VelZero
PosHigh             =    VelHigh
PosOscil            =    VelOscil
VelCtrlLoopFault    =    (TachoWireDisc or BrakeFailedON or EndSwitchPosDisc or
                         EndSwitchNegDisc ),
                         "Manual assignment of faults that appear inside the
                         velocity control loop"
PosCtrlLoopFault    =    ( VelCtrlLoopFault or VposWireDisc or VnegWireDisc or
                         VoutWireDisc or VelRefWireDisc ),
                         "Manual assignment of faults that appear inside the
                         position control loop"
```

Fault detection and isolation:

```
DetVelPos           =    ((PosMeasLow and VelMeasZero) or (PosMeasLow and
                         VelMeasHigh) or (PosMeasConst and VelMeasLow) or
                         (PosMeasConst and VelMeasHigh) or (PosMeasHigh and
                         VelMeasLow) or (PosMeasHigh and VelMeasZero) or
                         PosOscil),
```

```
                        "Detector:  Detect deviations between velocity and
                        position measurements based on model of gear and
                        integrator"
DetVelRefVel        =   ((VelRefDesLow and VelMeasZero) or (VelRefDesLow and
                        VelMeasHigh) or (VelRefDesZero and VelMeasLow) or
                        (VelRefDesZero and VelMeasHigh) or (VelRefDesHigh
                        and VelMeasLow) or (VelRefDesHigh and VelMeasZero) or
                        VelOscil),
                        "Detector:  Detect deviations between velocity reference
                        and velocity measur ement based on model of motor"
DetVelCur           =   ((CurLow and VelMeasZero) or (CurLow and VelMeasHigh) or
                        (CurZero and VelMeasLow) or (CurZero and VelMeasHigh) or
                        (CurHigh and VelMeasLow) or (CurHigh and VelMeasZero)),
                        "Detector:  Detect deviations between velocity
                        measurement and current measu rement based on model of
                        motor"
DetVelRefCur        =   ((CurLow and VelRefDesZero) or (CurLow and VelRefDesHigh)
                        or (CurZero and VelRefDesLow) or (CurZero and
                        VelRefDesHigh) or (CurHigh and VelRefDesLow) or (CurHigh
                        and VelRefDesZero) or CurOscil),
                        "Detector:  Detect deviations between velocity reference
                        and current based o n model of motor"
PosMeasFault        =   ( VposWireDisc or VnegWireDisc or VoutWireDisc ),
                        "Extra:  Position measurement faults"
VelCtrlFault        =   ( VelRefWireDisc or TachoWireDisc ),
                        "Extra:  Velocity controller faults"
OtherFault          =   ( BrakeFailedON or EndSwitchNegDisc or EndSwitchPosDisc
                        ),
                        "Extra:  Power drive, end-switch, and motor faults"
IsoPosMeasFault     =   (DetVelPos and not DetVelRefVel),
                        "Isolation:  Isolate position measurement fault"
IsoVelCtrlFault     =   (DetVelPos and DetVelRefVel),
                        "Isolation:  Isolate velocity measurement fault"
IsoOtherFault       =   (DetVelRefVel and not DetVelPos),
                        "Isolation:  Isolate power drive, end-switch, or motor
                        faults"
```

Decision logic:

```
PosEstim            =   IsoPosMeasFault,
                        "Supervisor:  If position measurement fault then enable
                        position estimation and use secondary position controller
                        #1"
VelCtrlBypass       =   IsoVelCtrlFault,
                        "Supervisor:  If velocity measurement fault then bypass
                        velocity controller by switching power drive to current
                        input and use secondary position contr oller #2"
CloseDown           =   IsoOtherFault,
                        "Supervisor:  If power drive fault, end-switch fault, or
                        motor fault then ma ke a shut-down"
```

Rules for mutually exclusive variables:

```
oneof(PosLow PosHigh PosConst PosOscil)
oneof(PosRefLow PosRefHigh PosRefConst)
oneof(PosMeasLow PosMeasHigh PosMeasConst PosMeasOscil)
oneof(VelMeasLow VelMeasHigh VelMeasZero VelMeasOscil)
oneof(VelRefDesLow VelRefDesHigh VelRefDesZero VelRefDesOscil)
oneof(VelRefLow VelRefHigh VelRefZero VelRefOscil)
oneof(CurRefLow CurRefHigh CurRefZero CurRefOscil)
oneof(CurLow CurHigh CurZero CurOscil)
oneof(VelLow VelHigh VelZero VelOscil)
```

```
oneof (VelRefWireDisc TachoWireDisc VposWireDisc VnegWireDisc VoutWireDisc
BrakeFailedON EndSwitchPosDisc EndSwitchNegDisc),
                        "Limit to single fault analysis.  One and only fault is
                        allowed true."
```

Appendix B

Beologic™ Rules for the Ørsted Satellite Case Study

This appendix contains listings of the Beologic™ Array Inference Toolbox (AIT) rules for the Ørsted satellite application.

B.1 Coil Driver Fault Propagation Graph

```
SignPlus          =   (NoSignFault and SignDesPlus)
SignMinus         =   (NoSignFault and SignDesMinus)
SignWrongSign     =   ((SignFailedPos and SignDesMinus) or (SignFailedNeg and
                      SignDesPlus))
ShuntVoltOK       =   (NoShuntFault and ShuntCurOK)
ShuntVoltZero     =   ((NoShuntFault and ShuntCurZero) or (ShuntShort))
ShuntVoltMax      =   ((NoShuntFault and ShuntCurMax) or (ShuntDisc))
CurAmplDesOK      =   ((not CurrentLoopFault and CurRefOK) or (CurrentLoopFault
                      and ShuntVoltOK))
CurAmplDesZero    =   ((not CurrentLoopFault and CurRefZero) or
                      (CurrentLoopFault and ShuntVoltMax))
CurAmplDesMax     =   ((not CurrentLoopFault and CurRefMax) or
                      (CurrentLoopFault and ShuntVoltZero))
SupplyOK          =   ((NoSwitchFault and CurAmplDesOK))
SupplyZero        =   ((NoSwitchFault and CurAmplDesZero) or T5Open)
SupplyMax         =   ((NoSwitchFault and CurAmplDesMax) or T5Short)
PowerUsageHigh    =   BridgeCurHigh
CurDesOK          =   (NoBridgeFault and not SignWrongSign and SupplyOK)
CurDesZero        =   ((NoBridgeFault and not SignWrongSign and SupplyZero)
                      or (T1Short and SignMinus) or (T2Short and SignPlus) or
                      (T1Open and SignPlus) or (T2Open and SignMinus) )
CurDesMax         =   (NoBridgeFault and not SignWrongSign and SupplyMax)
CurDesWrongSign   =   (NoBridgeFault and SignWrongSign)
BridgeCurHigh     =   ((NoBridgeFault and CurDrawnHigh) or (((T1Short and
                      SignMinus) or (T2Short and SignPlus)) and SupplyMax))
ShuntCurOK        =   ((NoBridgeFault and (CurReturnOK or CurReturnWrngSgn)) or
                      (((T1Short and SignMinus) or (T2Short and SignPlus)) and
                      SupplyOK))
```

```
ShuntCurZero       =   ((NoBridgeFault and CurReturnZero) or (T1Open and
                       SignPlus) or (T2Open and SignMinus) or (((T1Short and
                       SignMinus) or (T2Short and SignPlus)) and SupplyZero))
ShuntCurMax        =   ((NoBridgeFault and CurReturnMax) or (((T1Short and
                       SignMinus) or (T2Short and SignPlus)) and SupplyMax))
MagMomOK           =   ((NoMTQFault and CurDesOK) or (PosTermShortGND and
                       SignMinus and CurDesOK) or ( NegTermShortGND and SignPlus
                       and CurDesOK))
MagMomZero         =   (CurDesZero or (CoilShort or CoilDisc or (PosTermShortGND
                       and SignPlus) or (NegTermShortGND and SignMinus)))
MagMomMax          =   ((NoMTQFault and CurDesMax) or (PosTermShortGND and
                       SignMinus and CurDesMax) or (NegTermShortGND and SignPlus
                       and CurDesMax))
MagMomWrongSign    =   (CurDesWrongSign)
CurDrawnHigh       =   ((NoMTQFault or CoilShort or PosTermShortGND or
                       NegTermShortGND) and CurDesMax)
CurReturnOK        =   ((NoMTQFault and CurDesOK) or (CoilShort and CurDesOK))
CurReturnZero      =   (CurDesZero or CoilDisc or PosTermShortGND or
                       NegTermShortGND)
CurReturnMax       =   ((NoMTQFault and CurDesMax) or (CoilShort and CurDesMax))
CurReturnWrngSgn   =   (NoMTQFault and CurDesWrongSign)
oneof ( SignDesPlus SignDesMinus)
oneof ( SignPlus SignMinus SignWrongSign )
oneof ( CurRefOK CurRefZero CurRefMax )
oneof ( ShuntCurOK ShuntCurZero ShuntCurMax)
oneof ( ShuntVoltOK ShuntVoltZero ShuntVoltMax)
oneof ( CurAmplDesOK CurAmplDesZero CurAmplDesMax )
oneof ( CurDesOK CurDesZero CurDesMax CurDesWrongSign )
oneof ( SupplyOK SupplyZero SupplyMax)
oneof ( CurReturnOK CurReturnZero CurReturnMax CurReturnWrngSgn)
oneof ( MagMomOK MagMomZero MagMomMax MagMomWrongSign )
NoSignFault        =   (not (SignFailedPos or SignFailedNeg))
NoShuntFault       =   (not (ShuntDisc or ShuntShort))
NoSwitchFault      =   (not (T5Short or T5Open))
NoBridgeFault      =   (not ( T1Short or T2Short or T1Open or T2Open ))
NoMTQFault         =   (not ( PosTermShortGND or NegTermShortGND or CoilShort or
                       CoilDisc))
CurrentLoopFault   =   (ShuntShort or ShuntDisc or T1Open or T2Open or
                       T1Short or T2Short or T5Open or T5Short or CoilDisc or
                       PosTermShortGND or NegTermShortGND)
                       "Faults that affect the current control loop.  NB:
                       CoilShort, SignFailedPos, and SignFailedNeg are not
                       included as they do not affect the returned current."
CurrentMeasWrong   =   (not ShuntVoltOK)
oneof ( SignFailedPos SignFailedNeg ShuntShort ShuntDisc CoilShort CoilDisc
PosTermShortGND NegTermShortGND dummy)
oneof (not dummy T1Short T2Short T5Short T1Open T2Open T5Open)
```

B.2 Complete Ørsted Fault Propagation Graph

```
CDAXPowerHigh    =   (CDAXShuntShort or CDAXT5Open or CDAXPTermShort or
                     CDAXNTermShort)
CDAXMagMomZero   =   (CDAXShuntDisc or CDAXT5Short or CDAXCoilShort or
                     CDAXCoilDisc or CDAXPTermShort or CDAXNTermShort or
                     CDAXT1Short or CDAXT2Short or CDAXT1Open or CDAXT2Open)
CDAXMagMomMax    =   (CDAXShuntShort or CDAXT5Open or CDAXPTermShort or
                     CDAXNTermShort)
CDAXMagWngSgn    =   (CDAXFailedPos or CDAXFailedNeg)
```

```
CDAXCurMeasWng   =   (CDAXShuntShort or CDAXShuntDisc or CDAXT5Short or
                     CDAXT5Open or CDAXCoilDisc or CDAXPTermShort or
                     CDAXNTermShort or CDAXPTermShort or CDAXT1Open or
                     CDAXT2Open)
CDAYPowerHigh    =   (CDAYShuntShort or CDAYT5Open or CDAYPTermShort or
                     CDAYNTermShort)
CDAYMagMomZero   =   (CDAYShuntDisc or CDAYT5Short or CDAYCoilShort or
                     CDAYCoilDisc or CDAYPTermShort or CDAYNTermShort or
                     CDAYT1Short or CDAYT2Short or CDAYT1Open or CDAYT2Open)
CDAYMagMomMax    =   (CDAYShuntShort or CDAYT5Open or CDAYPTermShort or
                     CDAYNTermShort)
CDAYMagWngSgn    =   (CDAYFailedPos or CDAYFailedNeg)
CDAYCurMeasWng   =   (CDAYShuntShort or CDAYShuntDisc or CDAYT5Short or
                     CDAYT5Open or CDAYCoilDisc or CDAYPTermShort or
                     CDAYNTermShort or CDAYPTermShort or CDAYT1Open or
                     CDAYT2Open)
CDAZPowerHigh    =   (CDAZShuntShort or CDAZT5Open or CDAZPTermShort or
                     CDAZNTermShort)
CDAZMagMomZero   =   (CDAZShuntDisc or CDAZT5Short or CDAZCoilShort or
                     CDAZCoilDisc or CDAZPTermShort or CDAZNTermShort or
                     CDAZT1Short or CDAZT2Short or CDAZT1Open or CDAZT2Open)
CDAZMagMomMax    =   (CDAZShuntShort or CDAZT5Open or CDAZPTermShort or
                     CDAZNTermShort)
CDAZMagWngSgn    =   (CDAZFailedPos or CDAZFailedNeg)
CDAZCurMeasWng   =   (CDAZShuntShort or CDAZShuntDisc or CDAZT5Short or
                     CDAZT5Open or CDAZCoilDisc or CDAZPTermShort or
                     CDAZNTermShort or CDAZPTermShort or CDAZT1Open or
                     CDAZT2Open)
RandomMotion     =   ((PhaseStab and (SSA1Zero or SIMFault or CSCZero
                     or CDAXMagMomMax or CDAZMagMomMax or CDAXMagWngSgn
                     or CDAYMagWngSgn)) or CSCMax or CDAYMagMomMax or
                     CDAZMagWngSgn)
                     "FPA: Failure causes for random motion"
LargeError       =   ((PhaseStab and (SST1Fault or SST2Fault or SSA1High
                     or CDAXMagMomZero or CDAZMagMomZero)) or (!PhaseStab
                     and (CSCZero or CDAXMagMomMax or CDAZMagMomMax)) or
                     CDAYMagMomZero)
                     "FPA: Failure causes for large increase control errors"
SmallError       =   (!PhaseStab and (CDAXMagMomZero or CDAXMagWngSgn or
                     CDAXMagWngSgn or CDAYMagWngSgn or CDAYMagWngSgn or
                     CDAZMagMomZero))
                     "FPA: Failure causes for small increase in control
                     errors"
PowerHigh        =   (CDAXPowerHigh or CDAYPowerHigh or CDAZPowerHigh)
                     "FPA: Failure causes for increase in power consumption"
oneof (Dummy1 CSCZero CSCMax SSA1Zero SSA1High SST1Fault SST2Fault)
                     "SIMFault is not included in this list because it is
                     allowed together with other faults.  It determines the
                     setting of AttDetMode, which must be flexible."
oneof (!Dummy1 Dummy2 CDAXShuntShort CDAXShuntDisc CDAXT5Short CDAXT5Open
CDAXFailedPos CDAXFailedNeg CDAXCoilShort)
oneof (!Dummy2 Dummy3 CDAXCoilDisc CDAXNTermShort CDAXPTermShort CDAXT1Open
CDAXT1Short CDAXT2Open CDAXT2Short)
oneof (!Dummy3 Dummy4 CDAYShuntShort CDAYShuntDisc CDAYT5Short CDAYT5Open
CDAYFailedPos CDAYFailedNeg CDAYCoilShort)
oneof (!Dummy4 Dummy5 CDAYCoilDisc CDAYNTermShort CDAYPTermShort CDAYT1Open
CDAYT1Short CDAYT2Open CDAYT2Short)
oneof (!Dummy5 Dummy6 CDAZShuntShort CDAZShuntDisc CDAZT5Short CDAZT5Open
CDAZFailedPos CDAZFailedNeg CDAZCoilShort)
oneof (!Dummy6 CDAZCoilDisc CDAZNTermShort CDAZPTermShort CDAZT1Open CDAZT1Short
CDAZT2Open CDAZT2Short)
oneof ( RandomMotion LargeError SmallError )
```

B.3 Complete Rule base with FPG, FDI, Decision Logic, and Reconfiguration

Coil driver FPG:

```
CDAXPowerHigh     =   (CDAXShuntShort or CDAXT5Open or CDAXPTermShort or
                      CDAXNTermShort)
CDAXMagMomZero    =   (CDAXShuntDisc or CDAXT5Short or CDAXCoilShort or
                      CDAXCoilDisc or CDAXPTermShort or CDAXNTermShort or
                      CDAXT1Short or CDAXT2Short or CDAXT1Open or CDAXT2Open)
CDAXMagMomMax     =   (CDAXShuntShort or CDAXT5Open or CDAXPTermShort or
                      CDAXNTermShort)
CDAXMagWngSgn     =   (CDAXFailedPos or CDAXFailedNeg)
CDAXCurMeasWng    =   (CDAXShuntShort or CDAXShuntDisc or CDAXT5Short or
                      CDAXT5Open or CDAXCoilDisc or CDAXPTermShort or
                      CDAXNTermShort or CDAXPTermShort or CDAXT1Open or
                      CDAXT2Open)
CDAYPowerHigh     =   (CDAYShuntShort or CDAYT5Open or CDAYPTermShort or
                      CDAYNTermShort)
CDAYMagMomZero    =   (CDAYShuntDisc or CDAYT5Short or CDAYCoilShort or
                      CDAYCoilDisc or CDAYPTermShort or CDAYNTermShort or
                      CDAYT1Short or CDAYT2Short or CDAYT1Open or CDAYT2Open)
CDAYMagMomMax     =   (CDAYShuntShort or CDAYT5Open or CDAYPTermShort or
                      CDAYNTermShort)
CDAYMagWngSgn     =   (CDAYFailedPos or CDAYFailedNeg)
CDAYCurMeasWng    =   (CDAYShuntShort or CDAYShuntDisc or CDAYT5Short or
                      CDAYT5Open or CDAYCoilDisc or CDAYPTermShort or
                      CDAYNTermShort or CDAYPTermShort or CDAYT1Open or
                      CDAYT2Open)
CDAZPowerHigh     =   (CDAZShuntShort or CDAZT5Open or CDAZPTermShort or
                      CDAZNTermShort)
CDAZMagMomZero    =   (CDAZShuntDisc or CDAZT5Short or CDAZCoilShort or
                      CDAZCoilDisc or CDAZPTermShort or CDAZNTermShort or
                      CDAZT1Short or CDAZT2Short or CDAZT1Open or CDAZT2Open)
CDAZMagMomMax     =   (CDAZShuntShort or CDAZT5Open or CDAZPTermShort or
                      CDAZNTermShort)
CDAZMagWngSgn     =   (CDAZFailedPos or CDAZFailedNeg)
CDAZCurMeasWng    =   (CDAZShuntShort or CDAZShuntDisc or CDAZT5Short or
                      CDAZT5Open or CDAZCoilDisc or CDAZPTermShort or
                      CDAZNTermShort or CDAZPTermShort or CDAZT1Open or
                      CDAZT2Open)
```

Top level FPG:

```
RandomMotion      =   ((PhaseStab and (SSA1Low_ or SIMFault_)) or CSCFault_ or
                      CDAYMagMomMax_ or CDAZMagWngSgn_)
                      "FPA: Failure causes for random motion"
LargeError        =   ((PhaseStab and (SST1Fault_ or SST2Fault_ or SSA1High_))
                      or CDAXMagMomMax_ or CDAYMagMomZero_ or CDAZMagMomMax_)
                      "FPA: Failure causes for large increase control errors"
SmallError        =   (CDAXMagMomZero_ or CDAXMagWngSgn_ or CDAXMagWngSgn_ or
                      CDAYMagWngSgn_ or CDAYMagWngSgn_ or CDAZMagMomZero_)
                      "FPA: Failure causes for small increase in control
                      errors"
PowerHigh         =   (CDAXPowerHigh_ or CDAYPowerHigh_ or CDAZPowerHigh_)
                      "FPA: Failure causes for increase in power consumption"
```

Fault propagation through reconfiguration switches:

```
CDAXMagMomZero_   =   (CDAX and CDAEnabled and CDAXMagMomZero)
```

```
                    "FPA: Propagation of CDAX failed MZero"
CDAXMagMomMax_   =  (CDAX and CDAEnabled and CDAXMagMomMax)
                    "FPA: Propagation of CDAX failed MMax"
CDAXPowerHigh_   =  (CDAX and CDAEnabled and CDAXPowerHigh)
                    "FPA: Propagation of CDAX failed PMax"
CDAXMagWngSgn_   =  (CDAX and CDAEnabled and CDAXMagWngSgn)
                    "FPA: Propagation of CDAX failed wrong sign on mag.
                    mom."
CDAXCurMeasWng_  =  (CDAX and CDAEnabled and CDAXCurMeasWng)
                    "FPA: Propagation of CDAX failed wrong sign on current
                    meas."
CDAYMagMomZero_  =  (CDAY and CDAEnabled and CDAYMagMomZero)
                    "FPA: Propagation of CDAY failed MZero"
CDAYMagMomMax_   =  (CDAY and CDAEnabled and CDAYMagMomMax)
                    "FPA: Propagation of CDAY failed MMax"
CDAYPowerHigh_   =  (CDAY and CDAEnabled and CDAYPowerHigh)
                    "FPA: Propagation of CDAY failed PMax"
CDAYMagWngSgn_   =  (CDAY and CDAEnabled and CDAYMagWngSgn)
                    "FPA: Propagation of CDAY failed wrong sign on mag.
                    mom."
CDAYCurMeasWng_  =  (CDAY and CDAEnabled and CDAYCurMeasWng)
                    "FPA: Propagation of CDAY failed wrong sign on current
                    meas."
CDAZMagMomZero_  =  (CDAZ and CDAEnabled and CDAZMagMomZero)
                    "FPA: Propagation of CDAZ failed MZero"
CDAZMagMomMax_   =  (CDAZ and CDAEnabled and CDAZMagMomMax)
                    "FPA: Propagation of CDAZ failed MMax"
CDAZPowerHigh_   =  (CDAZ and CDAEnabled and CDAZPowerHigh)
                    "FPA: Propagation of CDAZ failed PMax"
CDAZMagWngSgn_   =  (CDAZ and CDAEnabled and CDAZMagWngSgn)
                    "FPA: Propagation of CDAZ failed wrong sign on mag.
                    mom."
CDAZCurMeasWng_  =  (CDAZ and CDAEnabled and CDAZCurMeasWng)
                    "FPA: Propagation of CDAZ failed wrong sign on current
                    meas."
SIMFault_        =  (AttDetModePrim and SIMFault)
                    "FPA: Propagation of SIM fault"
CSCFault_        =  (ADACEnabled and CSCFault)
                    "FPA: Propagation of CSC fault"
SSA1Low_         =  (!AttDetModePrim and SVectorEnabled and SSA1 and SSA1Low)
                    "FPA: Propagation of SSA1 failed low"
SSA1High_        =  (!AttDetModePrim and SVectorEnabled and SSA1 and
                    SSA1High)
                    "FPA: Propagation of SSA1 failed high"
SST1Fault_       =  (!AttDetModePrim and SST1 and SVectorEnabled and
                    SST1Fault)
                    "FPA: Propagation of SST1 fault"
SST2Fault_       =  (!AttDetModePrim and SST2 and SVectorEnabled and
                    SST2Fault)
                    "FPA: Propagation of SST2 failed"
```

Fault detectors:

```
DetSIMFault      =  SIMFault
                    "DET: Detection of SIM fault"
DetCSCFault      =  CSCFault
                    "DET: Detection of CSC fault"
DetSSA1Fault     =  ((SSA1 and SSA1Low) or (SSA1 and SSA1High))
                    "DET: Detection of SSA1 fault."
DetSST1          =  (SST1 and SST1Fault)
                    "DET: Detection of SST1 fault"
DetSST2          =  (SST2 and SST2Fault)
                    "DET: Detection of SST2 fault"
```

```
DetCDAXFault      =   (CDAX and CDAEnabled and (CDAXCurMeasWng or
                      ((CDAXMagWngSgn or CDAXMagMomZero) and !PhaseStab)))
                      "Detection of CDAZ fault.  "
DetCDAYFault      =   (CDAY and CDAEnabled and (CDAYCurMeasWng or
                      ((CDAYMagWngSgn or CDAYMagMomZero) and !PhaseStab)))
                      "Detection of CDAZ fault."
DetCDAZFault      =   (CDAZ and CDAEnabled and (CDAZCurMeasWng or
                      ((CDAZMagWngSgn or CDAZMagMomZero) and !PhaseStab)))
                      "Detection of CDAZ fault."
```

Decision logic:

```
!CDAX             =   DetCDAXFault
                      "SUP: Disable CDAY if failed"
!CDAY             =   DetCDAYFault
                      "SUP: Disable CDAY if failed"
!CDAZ             =   DetCDAZFault
                      "SUP: Disable CDAZ if failed"
CDAE_All_Off      =   ((!CDAX) and (!CDAY) and (!CDAZ))
                      "SUP: Flag:  CDAE_All_Off is true if no CDs in group one
                      is ON"
CDAEnabled        =   (ADACEnabled and !CDAE_All_Off)
                      "SUP: Enable coil group one if at least one CD is ON"
!ADACEnabled      =   DetCSCFault
                      "SUP: Enable control loop only if CSC is OK"
!AttDetModePrim   =   (PhaseStab and DetSIMFault)
!SSA1             =   (PhaseStab and DetSSA1Fault)
!SST1             =   (PhaseStab and DetSST1)
                      "SUP: Disable SST1 if it failed"
!SST2             =   (PhaseStab and DetSST2)
                      "SUP: Disable SST2 if it failed"
SVectorEnabled    =   (SST1 and SST2 and SSA1)
                      "SUP: Enable SS in EKF if satellite in Sun, all SST
                      works, and full 8 SSs are available"
```

One and only one fault:

```
oneof (Dummy1 CSCFault SSA1Low SSA1High SST1Fault SST2Fault)
                      "SIMFault is not included in this list because it is
                      allowed together with other faults.  It determines the
                      setting of AttDetMode, which must be flexible "
oneof (!Dummy1 Dummy2 CDAXShuntShort CDAXShuntDisc CDAXT5Short CDAXT5Open
CDAXFailedPos CDAXFailedNeg CDAXCoilShort)
oneof (!Dummy2 Dummy3 CDAXCoilDisc CDAXNTermShort CDAXPTermShort CDAXT1Open
CDAXT1Short CDAXT2Open CDAXT2Short)
oneof (!Dummy3 Dummy4 CDAYShuntShort CDAYShuntDisc CDAYT5Short CDAYT5Open
CDAYFailedPos CDAYFailedNeg CDAYCoilShort)
oneof (!Dummy4 Dummy5 CDAYCoilDisc CDAYNTermShort CDAYPTermShort CDAYT1Open
CDAYT1Short CDAYT2Open CDAYT2Short)
oneof (!Dummy5 Dummy6 CDAZShuntShort CDAZShuntDisc CDAZT5Short CDAZT5Open
CDAZFailedPos CDAZFailedNeg CDAZCoilShort)
oneof (!Dummy6 CDAZCoilDisc CDAZNTermShort CDAZPTermShort CDAZT1Open CDAZT1Short
CDAZT2Open CDAZT2Short)
```

Bibliography

Åström, K.J. (1991*a*). Assessment of Achievable Performance of Simple Feedback Loops. *Int. Journal of Adaptive Control and Signal Processing* **5**, 3–19.

Åström, K.J. (1991*b*). Intelligent Control. In proc.: *European Control Conference, ECC'91*. pp. 2328–2339.

Åström, K.J. and T. Hägglund (1995). *PID Controllers: Theory, Design, and Tuning*. 2 ed. Instrument Society of America, Research Triangle Park, NC.

Atkinson, R.M., M.R. Montakhab, D.J. Woollons, P.A. Hogan, C.R. Burrows and K.A. Edge (1993). DESHC: A Diagnostic Expert System for Hydraulic Circuits. In proc.: *CERT ONERA Tooldiag'93*. Vol. 3. pp. 882–895.

Avizienis, A. (1997). Toward Systematic Design of Fault-Tolerant Systems. *Computer* **30**(4), 51–58.

Bak, T. (1996). Onboard Attitude Determination for a Small Satellite. In proc.: *3rd ESA International Conference on Spacecraft Guidance, Navigation and Control Systems*. Noordwijk, the Netherlands.

Balls, B.V., G.R. Duke and A.B. Rentcome (1988). Application of RAM to the Design of Fault-Tolerant Safety Systems for Industrial Processes. In proc.: *Workshop on Reliability, Availability, and Maintainability on Industrial Process Control*. pp. 27–31.

Barbour, G.L. (1977). Failure Mode and Effects Analysis by Matrix Method. In proc.: *Annual Reliability & Maintainability Symposium*. pp. 114–119.

Basseville, M. (1988). Detecting Changes in Signals and Systems - A Survey. *Automatica* **24**(3), 309–326.

Basseville, M. and I.V. Nikiforov (1993). *Detection of Abrupt Changes: Theory and Application*. Prentice Hall, New Jersey.

Baumgarten, Götz (1996). *Ein Neues Flugregelungskonzept zur Rekonfiguration bei Stellgliedfehlern*. Forschungsbericht 96-22. Deutsche Forschungsanstallt für Luft- und Raumfahrt e.V. (DLR). Köln, Germany.

Beard, R.V. (1971). Failure Accommodation in Linear Systems Through Self-Reorganization. PhD thesis. Man Vehicle Laboratory. Cambridge, Massachusetts. Rept. MVT-71-1.

Bell, T.E. (1989). Managing Murphy's Law: Engineering a Minimum-Risk System. *Spectrum* pp. 24–27.

Benítez-Pérez, H., H.A. Thompson and P.J. Flemming (1997). Implementation of a Smart Sensor Using Analytical Redundancy Techniques. In proc.: *IFAC Safeprocess'97*. pp. 585–590.

Beologic (1996). *Beologic® visualSTATE™ 3.0 Classic for MS-Windows*.

Biering-Sørensen, S., F.O. Hansen, S. Klim and P.T. Madsen (1990). *Håndbog i Struktureret Program-Udvikling*. Teknisk Forlag. In Danish.

Blanke, M. (1996). Consistent Design of Dependable Control Systems. *Control Engineering Practice* **4**(9), 1305–1312.

Blanke, M. and P.B. Nielsen (1990). The Marine Engine Governor. In proc.: *Second International Conference on Maritime Communications and Control*. The Institute of Marine Engineers. London, UK. pp. 11–19.

Blanke, M. and R.B. Jørgensen (1995). Fault Handling Design for Integrated Marine Systems. In proc.: *3rd IFAC Workshop on Control Applications in Marine Systems*. Trondheim. pp. 86–95.

Blanke, M. and R.J. Patton (1995). Industrial Actuator Benchmark for Fault Detection and Isolation. *Control Engineering Practice* **3**(12), 1727–1730.

Blanke, M., R.I. Zamanabadi, S.A. Bøgh and C.P. Lunau (1997). Fault-Tolerant Control Systems - A Holistic View. *Control Engineering Practice* **5**(5), 693–702.

Blanke, M., S. Bøgh and R.B. Jørgensen (1993). Fault accommodation in feedback control systems. In: *Hybrid Systems* (R.L. Grossman, A. Nerode, A.P. Ravn and H. Rischel, Eds.). pp. 393–425. Lecture Notes in Computer Science (736). Springer Verlag.

Blanke, M., S. Bøgh, R.B. Jørgensen and R.J. Patton (1995). Fault Detection for Diesel Engine Actuator - A Benchmark for FDI. *Control Engineering Practice* **3**(12), 1731–1740.

Bøgh, S. (1995). Multiple Hypothesis-Testing Approach to FDI for the Industrial Actuator Benchmark. *Control Engineering Practice* **3**(12), 1763–1768.

Bøgh, S., M. Blanke and R.B. Jørgensen (1993). Nonlinear Characterization and Simulation of Industrial Position Control Loop. Technical Report R93-4017. Dept. of Control Eng., Aalborg University, Denmark.

Bøgh, S.A., R. Wisniewski and T. Bak (1997). Autonomous Attitude Control System for the Ørsted Satellite. In proc.: *Workshop on Control of Small Spacecraft*. Colorado, U.S.

Bøgh, S.A., R.I. Zamanabadi and M. Blanke (1995). Onboard Supervisor for the Ørsted Satellite Attitude Control System. In proc.: *5th ESA/ESTEC workshop on Artifical Intelligence and Knowledge Based Systems for Space*. pp. 137–152.

Brown, G.M., D.E. Bernard and R.D. Rasmussen (1995). Attitude and Articulation Control for the Cassini Spacecraft: A Fault Tolerance Overview. In proc.: *AIAA/IEEE 14th Digital Avionics Systems Conference*. pp. 184–192.

Calado, J.M.F. and P.D. Roberts (1997). Fault Detection and Diagnosis Based on Fuzzy Qualitative Reasoning. In proc.: *IFAC Safeprocess'97*. pp. 534–539.

Cassandras, C.G., S. Lafortune and G.J. Olsder (1995). Introduction to the modelling, control and optimization of discrete event systems. In: *Trends in Control* (A. Isidori, Ed.). pp. 217–291. Springer Verlag.

Cassar, J.Ph., M. Staroswiecki and P. Declerck (1994). Structural Decomposition for Large Scale Systems for the Design of Failure Detection and Identification Procedures. *Systems Science* **20**(1), 31–42.

Celi, R., C.Y. Huang and I.-C. Shih (1996). Reconfigurable Flight Control Systems for a Tandem Rotor Helicopter. In proc.: *52nd Annual Forum*. American Helicopter Society. Washington DC. pp. 1569–1588.

Cho, K.H. and J.T. Lim (1997). Fault-Tolerant Supervisory Control of Discrete Event Dynamical Systems. *Int. Journal of Systems Science* **28**(10), 1001–1009.

Chow, E.Y. and A.S. Willsky (1984). Analytical Redundancy and the Design of Robust Failure Detection Systems. *Transactions on Automatic Control* **AC-29**(7), 603–614.

David, R. and H. Alla (1992). *Petri Nets and Grafcet: Tools for Modelling Discrete Event Systems*. Prentice Hall.

de Selding, P.D. (1996). Faulty software caused ariane 5 failure. Published in Space News 24 June 1996.

Ding, X. and P.M. Frank (1991). Frequency Domain Approach and Threshold Selector for Robust Model-based Fault Detection and Isolation. In proc.: *IFAC Safeprocess'91*. Vol. 1. pp. 307–312.

DoD (1980). Procedure for Performing a Failure Mode, Effects and Criticality Analysis. Technical report. Dept. of Defence, NAEC. Lakehurst, NJ, US.

Douglas, R.K. and J.L. Speyer (1996). Robust Fault Detection Filter Design. *Journal of Guidance, Control, and Dynamics* **19**(1), 214–218.

Drejer, N. (1994). Methods of Run-Time Error Detection in Distributed Process Control Software. PhD thesis. Dept. of Control Eng., Aalborg University, Denmark.

Efe, M. and D.P. Atherton (1997). The IMM Approach to the Fault Detection Problem. In proc.: *System Identification SYSID'97*. pp. 625–630.

Emani-Naeini, A., M.M. Athter and S.M. Rock (1988). Effect of Model Uncertainty on Failure Detection: The Threshold Selector. *IEEE Trans. on Automatic Control* **33**(12), 1106–1115.

ESA-ECSS (1996*a*). Space Product Assurance: Dependability (ECSS-Q-30A). Technical report. European Space Agency ESA, Requirements & Standards Division. Noordwijk, The Netherlands.

ESA-ECSS (1996*b*). Space Product Assurance: Safety (ECSS-Q-40A). Technical report. European Space Agency ESA, Requirements & Standards Division. Noordwijk, The Netherlands.

ESA-ECSS (1997). Glossary of Terms (ECSS-P-001A, rev. 1). Technical report. European Space Agency ESA, Requirements & Standards Division. Noordwijk, The Netherlands.

Ford/General Motors/Chrysler (1995). Potential Failure Mode and Effects Analysis (FMEA) - Reference Manual. Technical report. Chrysler Corporation, Ford Motor Company, and General Motors Corporation.

Frank, P.M. (1990). Fault Diagnosis in Dynamic Systems using Analytical and Knowledge-Based Redundancy - A Survey and some New Results. *Automatica* **26**(3), 459–474.

Frank, P.M. (1995). Advances in Fault Tolerance by Model-Based Fault Diagnosis. In proc.: *ESF workshop on Control of Complex Systems COSY*. pp. 15–21.

Frank, P.M. (1996). Analytical and Qualitative Model-Based Fault Diagnosis - a Survey and Some New Results. *European Journal of Control* **2**(1), 6–28.

Frank, P.M. and J. Wunnenberg (1989). Robust fault diagnosis using unknown input observer schemes. In: *Fault Diagnosis in Dynamic Systems - Theory and Applications* (R. Patton, P.M. Frank and R. Clark, Eds.). Chap. 3. Prentice Hall International.

Frank, P.M. and X. Ding (1994). Frequency Domain Approach to Optimally Robust Residual Generation. *Automatica* **30**(5), 789–804.

Franksen, O.I. (1979). Group representation of finite polyvalent logic - a case study using APL notation. In: *A Link between Science and Applications of Automatic Control* (A. Niemi, Ed.). pp. 875–887. Pergamon Press, Oxford and New York.

Frydkjær, K. (1997). Failure of the Conveyer Belt Encoder in 1992. Personal contact to K. Frydkjær, Sander Hansen and H. Rosentoft, Carlsberg, Denmark.

Füssel, D., P. Ballé and R. Isermann (1997). Closed Loop Fault Diagnosis Based on a Nonlinear Process Model and Automatic Fuzzy Rule Generation. In proc.: *IFAC Safeprocess'97*. pp. 359–364.

Gao, Z. and P. Antsaklis (1991). Stability of the Pseudo-Inverse Method for Reconfigurable Control Systems. *Int. Journal of Control* **53**(3), 717–729.

García, E.A. and P.M. Frank (1996). On the Relationship between Observer Based and Parameter Identification Based Approaches to Fault Detection. In proc.: *13th IFAC World Congress*. Vol. N. San Francisco. pp. 25–30.

García, E.A. and P.M. Frank (1997). Deterministic Nonlinear Observer-Based Approaches to Fault Diagnosis: A Survey. *Control Engineering Practice* **5**(5), 663–760.

García, E.A., B.Köppen-Seliger and P.M. Frank (1995). A Frequency Domain Approach to Residual Generation for the Industrial Actuator Benchmark. *Control Engineering Practice* **3**(12), 1747–1750.

Gertler, J. (1993). Optimal Residual Decoupling for Robust Fault Diagnosis. In proc.: *CERT ONERA Tooldiag'93*. pp. 436–452.

Gertler, J. (1997*a*). Fault Detection and Isolation using Parity Relations. *Control Engineering Practice* **5**(5), 653–661.

Gertler, J. (1997*b*). On the Relationship between Parity Relations and Parameter Estimation. In proc.: *IFAC Safeprocess'97*. pp. 468–473.

Grainger, R.W., J. Holst, A.J. Isaksson and B.M. Ninness (1995). A Parametric Statistical Approach to FDI for the Industrial Actuator Benchmark. *Control Engineering Practice* **3**(12), 1757–1762.

Hägglund, T. (1995). A Control-Loop Performance Monitor. *Control Engineering Practice* **3**, 1543–1551.

Hägglund, T. and K.J. Åström (1997). Supervision of Adaptive Control Algorithms. In proc.: *IFAC Conference on Control of Industrial Systems*. Belfort, France.

Han, Z. and P.M. Frank (1997). Physical Parameter Estimation Based FDI with Neural Networks. In proc.: *IFAC Safeprocess'97*. pp. 294–299.

Hermans, F.J.J. and M.B. Zarrop (1997). Parameter Estimation Using Sliding Mode Principles. In proc.: *IFAC Safeprocess'97*. pp. 282–287.

Herrin, S.A. (1981). Maintainability Applications Using the Matrix FMEA Technique. *Transactions on Reliability* **R-30**(2), 212–217.

Hignett, K.C. (1996). *Practical Safety and Reliability Assessment*. 1 ed. E & FN Spoon. 2-6 Boundary Row, London SE1 8HN, UK.

Himmelblau, D.M., R.W. Barker and W. Suewatanakul (1991). Fault Classification with Aid of Artificial Neural Networks. In proc.: *IFAC Safeprocess'91*. Vol. 2. pp. 369–373.

Höfling, Th., Th. Pfeufer, R. Deibert and R. Isermann (1995). An observer and Signal Processing Approach to FDI for the Industrial Actuator Benchmark Test. *Control Engineering Practice* **3**(12), 1741–1746.

Holding, D. (1991). Software fault tolerance. In: *Failsafe Control Systems, Application and Emergency Management* (K. Warwick and M.T. Tham, Eds.). Chap. 2. Chapman and Hall.

Hou, M. and P.C. Müller (1994). Fault-Detection and Isolation Observers. *Int. Journal of Control* **60**(5), 827–846.

Huang, C.Y. and R.F. Stengel (1990). Restructurable Control Using Proportional-Integral Implicit Model Following. *J. Guidance* **13**(2), 303–309.

Isaksson, A.J. (1993). An On-line Threshold Selector for Failure Detection. In proc.: *CERT ONERA Tooldiag'93*. pp. 628–634.

Isermann, R. (1984). Process Fault-Detection based on Modelling and Estimation Methods - A Survey. *Automatica* **20**(4), 387–404.

Isermann, R. (1993). Fault Diagnosis of Machines via Parameter Estimation and Knowledge Processing. *Automatica* **29**, 815–836.

Isermann, R. (1997). Supervision, Fault-Detection and Fault-Diagnosis Methods - an Introduction. *Control Engineering Practice* **5**(5), 639–652.

Isermann, R. and P. Ballé(1997). Trends in the Application of Model-Based Fault Detection and Diagnosis of Technical Processes. *Control Engineering Practice* **5**(5), 709–719.

Isermann, R., H. Keller and U. Raab (1993). Smart Actuators. In proc.: *CERT ONERA Tooldiag'93*. pp. 117–126.

Jensen, B., L. Nielsen and M. Svavarsson (1994). Approach for design of reliable controller. Master's thesis. Dept. of Control Eng., Aalborg University, Denmark. In Danish.

Jones, H.L. (1973). Failure Detection in Linear Systems. PhD thesis. Dept. of Aeronautics and Astronautics, M.I.T.. Cambridge, Massachusetts.

Jørgensen, R.B. (1995). Development and Test of Methods for Fault Detection and Isolation. PhD thesis. Dept. of Control Eng., Aalborg University, Denmark.

Jørgensen, R.B., R.J. Patton and J. Chen (1995). An Eigenstructure Assignment Approach to FDI for the Industrial Actuator Benchmark Test. *Control Engineering Practice* **3**(12), 1751–1756.

Kapur, K.C. (1977). *Reliability in Engineering Design*. John Wiley & Sons.

Keller, J.Y. and M. Darouach (1997). A New Estimator for Dynamic Stochastic Systems with Unknown Inputs: Application to Robust Fault Diagnosis. In proc.: *IFAC Safeprocess'97*. pp. 177–180.

Kiupel, N. and P.M. Frank (1997). A Fuzzy FDI Decision Making System for the Support of the Human Operator. In proc.: *IFAC Safeprocess'97*. pp. 731–736.

Köppen-Seliger, B. and P.M. Frank (1995). Fault Detection and Isolation in Technical Processes with Neural Networks. In proc.: *Decision and Control CDC'95*. New Orleans. pp. 2414–2419.

Köppen-Seliger, B. and P.M. Frank (1996). Neural Network in Model-Based Fault Diagnosis. In proc.: *13th IFAC World Congress*. 7f-02 5. pp. 67–72.

Kovio, H.N. (1994). Artificial Neural Networks in Fault Diagnosis and Control. *Control Engineering Practice* **2**(7), 89–101.

Laprie, J.C. (1987). Computing Systems Dependability and Fault Tolerance: Basic Concepts and Terminology. *Agardograph* (289), 1.1–1.15.

Lauber, R.J. (1991). Aspects of Achieving Total Systems Availability. In proc.: *IFAC Safeprocess'91*. Vol. 1. pp. 35–41.

Lee, K.-S. and J. Vagners (1997). Reliable Decision Unit utilizing Fuzzy Logic for Observer Based Fault Detection Systems. In proc.: *IFAC Safeprocess'97*. pp. 693–698.

Lee, S.C. and A.G. Santo (1996). Near Earth Asteroid Rendezvous (NEAR) Spacecraft Safing Design. *Acta Astronautica* **39**(1-4), 197–206.

Legg, J.M. (1978). Computerized Approach for Matrix-Form FMEA. *Transactions on Reliability* **R-27**(1), 254–257.

Leitch, R. (1993). Engineering Diagnosis: Matching Problems to Solutions. In proc.: *CERT ONERA Tooldiag'93*. pp. 837–844.

Lions, J.L. (1996). Ariane 5 Flight 501 Failure. Inquiry board report. European Space Agency. Available at http://www.esrin.esa.it/htdocs/tidc/Press/Press96/ariane5rep.html.

Lunze, J. and F. Schiller (1996). Diagnosis Based on a Probabilistic Model of Dynamic Systems. In proc.: *IFAC World Congress*. pp. 145–150.

Lunze, J. and F. Schiller (1997). An Example of Fault Diagnosis by Means of Probabilistic Logic Reasoning. In proc.: *IFAC Safeprocess'97*. pp. 540–545.

Lutz, R.R. and R.M. Woodhouse (1996). Contributions of SFMEA to Requirements Analysis. In proc.: *IEEE International Conference on Requirements Engineering, ICRE*. Colorado Springs, CO, USA. pp. 44–51.

Mazza, C., A. Scheffer, B. Melton, D. de Pablo, R. Stevens and J. Fairclough (1992). ESA's Software Engineering Standard - The Foundation for Reliable Software. *ESA Bulletin* **69**, 97–104.

Mediavilla, M., L.J. Miguel and P. Vega (1997). Isolation of Multiplicative Faults in the Industrial Actuator Benchmark. In proc.: *IFAC Safeprocess'97*. pp. 855–860.

Miguel, L.J., M. Mediavilla and J.R. Perán (1997). Decision-Making Approaches for a Model-Based FDI Method. In proc.: *IFAC Safeprocess'97*. pp. 719–725.

Misra, A. (1994). Sensor-Based Diagnosis of Dynamical Systems. PhD thesis. Vanderbilt University, US.

Moerder, D.D., N. Halyo, J.R. Broussard and A.K. Caglayan (1989). Application of Precomputed Control Laws in a Reconfigurable Aircraft Flight Control System. *J. Guidance* **12**(3), 325–333.

Møller, G. (1995). On the Technology of Array-based Logic. PhD thesis. Electric Power Eng. Dept., Tech. University of Denmark, Lyngby, Denmark.

Montmain, J. and S. Gentil (1993). Decision-Making in Fault Detection: A Fuzzy Approach. In proc.: *CERT ONERA Tooldiag'93*. pp. 653–660.

Morse, A.S. (1996). Supervisory Control of Families of Linear Set-Point Controllers - Part 1: Exact Matching. *IEEE Trans. on Automatic Control* **41**(10), 1413–1431.

Morse, W.D. and K.A. Ossman (1990). Model Following Reconfigurable Flight Control System for the AFTI/F-16. *J.Guidance* **13**(6), 969–976.

NASA (1993). NASA Safety Policy and Requirements Document. NASA handbook, nhb 1700.1 (v1-b). QS Safety & Risk Management Division. Available at http://nodis.hq.nasa.gov/Library/Directives/NASA-WIDE/Procedures/Organization_and_Administration/N_HB_1700_1_(V1-B).html.

Nelson, V.P. (1990). Fault-Tolerant Computing: Fundamental Concepts. *Computer* **23**(7), 19–25.

Nielsen, S.B., R. Patton, M. Blanke and R.B. Jørgensen (1993). Industrial Actuator Benchmark Test. Technical Report R93-4020. Aalborg University, Denmark and York University, England.

Nikhoukhah, R.N. (1994). Innovations Generation in the Presence of Unknown Inputs: Application to Robust Failure Detection. *Automatica* **30**, 1851–1868.

Nilsen, S.O. and M. Blanke (1996). Overall Specification FMEA Module. Technical Report R96-4149. Dept. of Control Eng., Aalborg University, Denmark.

Ochi, Y. and K. Kanai (1991). Design of Restructurable Flight Control Systems Using Feedback Linearization. *Journal of Guidance, Control & Dynamics* **14**(5), 903–911.

Ostroff, A.J. (1985). Techniques for Accommodating Control Effector Failures on Mildly Statically Unstable Airplane. In proc.: *American Control Conference*. pp. 906–913.

Overkamp, A. (1997). Supervisory Control Using Partial Failure Semantics and Partial Specifications. *IEEE Trans. on Automatic Control* **42**(4), 498–510.

Patton, R.J. (1997). Fault-Tolerant Control: The 1997 Situation. In proc.: *IFAC Safeprocess'97*. pp. 1033–1055.

Patton, R.J. and J. Chen (1991*a*). A Review of Parity Space Approaches to Fault Diagnosis. In proc.: *IFAC Safeprocess'91*. Vol. 1. pp. 239–255.

Patton, R.J. and J. Chen (1991*b*). Robust Fault Detection using Eigenstructure Assignment: A Tutorial Consideration and some new Results. In proc.: *30th conf. on Decision and Control, CDC'91*. pp. 2242–2247.

Patton, R.J. and J. Chen (1996). Robust fault detection and isolation (FDI) systems. In: *Techniques in Discrete and Continuous Robust Systems* (C.T. Leondes, Ed.). Vol. 74 of *Dynamics & Control Series*. pp. 171–224. Academic Press.

Patton, R.J. and S.M. Kangethe (1989). Robust fault diagnosis using eigenstructure assignment of observers. In: *Fault Diagnosis in Dynamic Systems - Theory and Applications* (R. Patton, P.M. Frank and R. Clark, Eds.). Chap. 4. Prentice Hall International.

Payton, D.W., D. Keirsey, D.M. Kinble, J. Krozel and K. Rosenblatt (1992). Do Whatever Works: A Robust Approach to Fault-Tolerant Autonomous Control. *Journal of Applied Intelligence* **2**, 225–250.

Pell, B., D.E. Bernard, S.A. Chien, E. Gat, N. Muscettola, P.P. Nayak, M.D.Wagner and B.C. Williams (1997). An Autonomous Spacecraft Agent Prototype. In proc.: *1st Int. Conf. on Autonomous Agents*. pp. 253–261.

Pressman, R.S. (1988). *Software Engineering - A Practitioner's Approach*. McGraw-Hill Book Co.

Price, C.J., D.R. Pugh, N. Snooke, J.E. Hunt and M.S. Wilson (1997). Combining Functional and Structural Reasoning for Safety of Electrical Designs. *The Knowledge Engineering Review* **12**, 271–287.

Rauch, H.E. (1995). Autonomous Control Reconfiguration. *IEEE Control System Magazine* **15**(6), 37–48.

Rotstein, G.E., A. Sanchez and N. Alsop (1995). Synthesis of Procedural Controllers and the Automatic Generation of Sequential Controllers. In proc.: *Workshop on Analysis and Design of Event-Driven Operation in Process Systems (ADEDOPS)*. Centre for Process Systems Engineering, Imperial College, UK.

Santo, A.G., S.C. Lee and R.E. Gold (1995). NEAR Spacecraft and Instrumentation. *The Journal of Astronautical Sciences* **43**(4), 373–397.

Shafaghi, A., P.K. Andow and F.P. Lees (1984). Fault Tree Synthesis based on Control Loop Structure. *Chemical Engineering Research and Design* **62**, 101–110.

Slonski, J.P. (1996). System Fault Protection Design for the Cassini Spacecraft. In proc.: *Aerospace Applications Conference*. IEEE. Aspen, CO. pp. 279–292.

Tzafestas, S. and K. Watanabe (1990). Modern Approaches to System/Sensor Fault Detection and Diagnosis. *Journal A.* **31**(4), 42–57.

Veillette, R.J., J.V. Medani and W.R. Perkins (1992). Design of Reliable Control Systems. *Trans. on Automatic Control* **37**(3), 290–304.

Verbruggen, H., S. Tzafestas and E. Zanni (1995). Knowledge-based fault diagnosis of technical systems. In: *Artificial Intelligence in Industrial Decision Making, Control and Automation* (S.G. Tzafestas and H.B. Verbruggen, Eds.). pp. 449–506. Kluwer Academic.

Visinsky, M.L., J.R. Cavallaro and I.D. Walker (1994). Robotic Fault Detection and Fault Tolerance: A Survey. *Reliability Engineering and System Safety* **46**, 139–158.

Walker, B.K. and K.Y Huang (1995). FDI by Extended Kalman Filter Parameter Estimation for Industrial Actuator Benchmark. *Control Engineering Practice* **3**(12), 1769–1774.

Willsky, A.S. (1976). A Survey of Design Methods for Failure Detection in Dynamic Systems. *Automatica* **12**, 601–611.

Wimmer, W. (1997). Lessons to be Learned from European Science and Application Space Missions. *Control Engineering Practice* **5**(2), 155–165.

Wisniewski, R. (1996). Satellite Attitude Control Using Only Electromagnetic Actuation. PhD thesis. Dept. of Control Eng., Aalborg University, Denmark.

Wonham, W.M. (1988). A control theory for discrete-event system. In: *Advanced Computing Concepts and Techniques in Control Engineering* (M.J. Denham and A.J. Laub, Eds.). pp. 129–169. Springer-Verlag.

Yang, J.C. and D.W. Clarke (1997). The Self-Validating Actuator. In proc.: *IFAC Safeprocess'97*. pp. 579–584.

Yang, J.E., S.H. Han, J.H. Park and Y.H. Jin (1997). Analytic Method to Break Logical Loops Automatically in PSA. *Reliability Engineering and System Safety* **56**, 101–105.

Yazdi, H. (1997). Control and Supervision of Event-Driven Systems. PhD thesis. Technical University of Denmark.

Zamanabadi, R.I., S.A.Bøgh and M. Blanke (1996). On the Design and Realization of Supervisory Functions in Fault Tolerant Control. In proc.: *KoREMA conference*. Zagreb, Croatia. pp. 91–95.

Zheng, C., R.J. Patton and J. Chen (1997). Robust Fault-Tolerant Systems Synthesis via LMI. In proc.: *IFAC Safeprocess'97*. pp. 347–352.