Effekter på systemsäkerhet orsakade av diagnossystem

Examensarbete
utfört vid **Fordonssystem**

av
Anders Holmstrand

Reg nr: LiTH-ISY-EX-3536-2003

20 november 2003

Effekter på systemsäkerhet orsakade av diagnossystem

Examensarbete

utfört vid **Fordonssystem**,
Institutionen för systemteknik
på **Linköpings universitet**

av **Anders Holmstrand**

Reg nr: LiTH-ISY-EX-3536-2003

Handledare: **Jonas Biteus**
Linköpings universitet

Examinator: **Erik Frisk**
Linköpings universitet

Linköping, 20 november 2003

Avdelning, Institution
Division, Department

Fordonssystem,
Institutionen för systemteknik
581 83 Linköping

Datum
Date

20 november 2003

Språk
Language
☒ Svenska/Swedish
☐ Engelska/English
☐ _____

Rapporttyp
Report category
☐ Licentiatavhandling
☒ Examensarbete
☐ C-uppsats
☐ D-uppsats
☐ Övrig rapport
☐ _____

ISBN
—

ISRN
LITH-ISY-EX-3536-2003

Serietitel och serienummer **ISSN**
Title of series, numbering —

URL för elektronisk version
http://www.vehicular.isy.liu.se
http://www.ep.liu.se/exjobb/isy/2003/3536/

Titel Effekter på systemsäkerhet orsakade av diagnossystem

Title Effects on system safety caused by diagnostic systems

Författare Anders Holmstrand
Author

Sammanfattning
Abstract

Adding a diagnostic and supervisorial system to a technical process can theoretically improve the system safety. However, due to the lack of proper methods, not all the potential effects of a diagnostic system is included in the analysis. This thesis focuses on the possibility that the adding of a diagnostic system in itself may cause negative safety effects. Quality analysis techniques have been used as tools for achieving the goals. The two most important being *Failure Mode and Effects Analysis (FMEA)* and *Fault Tree Analysis (FTA)*. A working method using FMEA and FTA is presented and tested on a suitable project. It can be shown that a diagnostic system adds negative system safety effects. Therefore, it is the recommendation of this thesis to use the proposed principles when dealing with system safety analysis including diagnostic systems.

Nyckelord
Keywords Systemsäkerhet, kvalitet, riskanalys, diagnos, FMEA, FTA

Sammanfattning

Genom att införa diagnos- och övervakningssystem i tekniska processer kan man teoretiskt visa på förbättrade säkerhetsegenskaper hos det totala systemet. Eftersom det saknas riktlinjer för hur detta arbete ska ske, undersöks inte effekterna som uppkommer totalt sett. I denna rapport undersöks om ett diagnossystem kan försämra den totala systemsäkerheten. För ändamålet används kvalitetstekniska metoder. De två mest centrala metoderna FMEA och FTA bildar tillsammans en grund för analyser av de effekter som ett diagnossystem kan ha på systemsäkerheten för det totala systemet. En arbetsgång för analys baserad på dessa metoder presenteras och testas på ett befintligt projekt. Det går att påvisa att ett diagnossystem inför osäkerheter som påverkar den totala systemsäkerheten. Det är därför en rekommendation att arbeta enligt de principer som presenteras i denna rapport.

Nyckelord: Systemsäkerhet, kvalitet, riskanalys, diagnos, FMEA, FTA

Tack

Under arbetets gång har jag fått hjälp från alla möjliga håll med allt från matematiska funderingar till moraliskt stöd. Jag vill tacka avdelningen för fordonssystem och all personal där för att de gjorde arbetet möjligt. Speciellt min handledare Jonas Biteus som trodde på projektet och som manade på mig. Jan-Åke Larsson ska ha en stor eloge för sin hjälp med miktex och för att han agerat bollplank. Detsamma gäller Petter Isaksson som är en klippa på att lyssna till beklagan och frustrationer. Joakim Roubert tackas för utlånat material. Det världsomspännande datornätet Internet ska även det ha tack för stöd och uppmuntran. Till sist vill jag tacka min sambo Linnea för att hon alltid är ett ljus i min tillvaro.

Innehåll

Sammanfattning **v**

Tack **vi**

1 Inledning **1**
1.1 Bakgrund 1
1.2 Mål 2
1.3 Avgränsningar 2
1.4 Framtida arbete 3

2 Systemsäkerhet och kvalitetstekniska metoder **4**
2.1 Tillförlitlighet samt diagnos- och övervakningssystem . . 4
2.1.1 Modellbaserad diagnos och övervakning 5
2.1.2 Reparabla och irreparabla system 6
2.2 Systemsäkerhet 6
2.3 Introduktion till FMEA 7
2.3.1 Process-FMEA 8
2.3.2 Konstruktions-FMEA och FMECA 8
2.3.3 Arbetsgång för FMEA 9
2.3.4 Förutsättningar 9
2.3.5 Slutsatser från en FMEA 10
2.4 Felträdsanalys - FTA 11
2.4.1 Kvalitativ FTA 12
2.4.2 Minimala cut sets 12
2.4.3 Kvantitativ FTA 14
2.4.4 Sannolikhetsberäkningar 14
2.4.5 Arbetsgång 15
2.5 Felsannolikheter och felintensiteter 16

3 Felbenägenhet hos diagnossystem **18**
3.1 Vad bygger upp ett diagnos- och övervakningssystem? . 18
3.2 Felbenägenhet hos komponenter 19
3.3 Bedömning av felbenägenhet i datorarkitektur 19

3.4 Bedömning av fel i diagnospresentation 19
3.5 Analys av felrisk i mjukvarukod 20
3.5.1 Cyklomatisk komplexitet - McCabes teorem . . . 20
3.5.2 Halsteads metod, 4-tupelmetoden 21
3.5.3 KLOC 22
3.5.4 Fault seeding och fault injection 23
3.5.5 Slutsatser om kodanalys 23

4 Utökning med diagnossystem **24**
4.1 Antagande om utgångspunkt 24
4.2 Kravspecifikation 25
4.2.1 Krav 1 - följ standarder 25
4.2.2 Krav 2 - generell metod 25
4.2.3 Krav 3 - återspegla effekterna tydligt 25
4.2.4 Krav 4 - enkelhet 25
4.3 FMEA som kvalitativ underlagsanalys 26
4.4 FTA som kvantitativ analysmetod 26
4.5 Varför denna modell för arbetsgång? 28
4.5.1 Krav 1 - följ standarder 28
4.5.2 Krav 2 - generell metod 28
4.5.3 Krav 3 - återspegla effekterna tydligt 28
4.5.4 Krav 4 - enkelhet 29
4.6 Systemsäkerhetsprojekt med bilbana som principsystem 29
4.6.1 Inledning 29
4.6.2 Principiella likheter mellan bilbana och flygplan 30
4.6.3 Analys av bilbana utan diagnos 30
4.6.4 Analys av bilbana med diagnos 30
4.6.5 Utökad analys av diagnossystemets effekter . . . 31
4.6.6 FMEA över diagnossystemets negativa effekter . 35
4.6.7 FTA över diagnossystemets negativa effekter . . 36

5 Diskussion **39**
5.1 Slutsatser 39

Referenser **40**

Kapitel 1

Inledning

Då moderna fordon tenderar att bli alltmer avancerade och allt fler komponenter och system är datorreglerade, har behovet av automatiserad systemövervakning ökat. Teorierna kring diagnos- och övervakningssystem är därför en allt viktigare del vid fordonskonstruktion. Fordonsbranschen har även en annan sida. Som bransch betraktat ligger man långt framme vad gäller kvalitetsarbete. Processtyrningen av projekt följer en väl dokumenterad standard för att uppnå en hög tillförlitlighet hos de färdiga produkterna. Detta kallas att man har en hög grad av *systemsäkerhet*. Dock saknas riktlinjer i processtyrningen för den relativt nya vetenskapen om diagnos- och övervakningssystem. Detta arbetet syftar till att undersöka effekterna av diagnos- och övervakningssystem samt föreslå riktlinjer för processtyrningen av denna vetenskap. Arbetet bedrivs med fokus på befintliga strategier inom industrin. Detta examensarbete har utförts vid institutionen för systemvetenskap, avdelningen för fordonssystem.

1.1 Bakgrund

Moderna fordon består av många delsystem och de flesta är dag styrda av tämligen avancerade datorsystem. Dessa system går ofta under den något subjektiva begreppet ”integrerat datorsystem”. Detta tolkas här som att en rad I/O-enheter samlar data som analyseras av en datorkärna. Denna står således för insamling av data, bearbetning och beslut. För att ge systemet en så god möjlighet som möjligt att fatta rätt beslut har man byggt in diagnos- och övervakningssystem. Dessa diagnossystem är konstruerade enligt matematiska principer som syftar till att möjliggöra en så felfri process som möjligt. Men blir det alltid så i verkligheten? Det finns ett stort gap mellan den vetenskapliga teorin om diagnossystem och sättet som industrin arbetar på. Det är önskvärt

att anpassa arbetet med diagnossystem till industriprocessernas verklighet och krav.

Ett spektakulärt exempel på ett fall där diagnossystemet motverkade sitt syfte är Ariane 5-projektet. 1996 skulle den Europeiska rymdstyrelsen sända iväg Ariane-5 på sin jungfrufärd. Det slutade med att raketens självförstörelsemekanism detonerade och skapade ett spektakulärt fyrverkeri. Kostnad: femhundra miljoner US dollar; då är utvecklingskostnaden på sju miljarder US dollar undantagen. Orsaken till misslyckandet var en självdiagnostisk del inom styrsystemet som hade försökt göra en konvertering av datatyper, men misslyckats. Det alstrade feldata som skickades till huvuddatorn. Bristande programvaruutveckling gjorde dock att huvuddatorn tolkade diagnosdatan som styrdata och riktade felaktigt om styrraketerna i horribla vinklar. Då styrsystemet och dess backupsystem samtidigt slutade fungera, fanns ingen möjlighet att reda upp situationen. Diagnossystemet införde säkerhet åt de komponenter det övervakade, men införde samtidigt en osäkerhet i sig själv; i detta fall en kritisk osäkerhet. [3]

1.2 Mål

Det saknas rutiner inom systemsäkerhetsarbete för hur man systematiskt ska arbeta med diagnossystem. Gapet mellan vetenskapen om diagnossystem och industrins verklighet är för stort. Man vill därför kunna införliva diagnosteknikernas arbete i det kvalitetsmässiga ingenjörsarbetet. Design och analys av ett diagnossystem ska så långt som det är möjligt följa samma arbetsgång och ha samma rutiner som gängse kvalitetsarbete. Resultatet av att skapa en enhetlig process blir en ökad förståelse för diagnossystemets inverkan, både på den enskilda komponenten/delsystemet såväl som på det totala systemet. Det införda diagnossystemet måste visa på signifikanta mervärdeseffekter för det totala systemet.

Målet är att identifiera de nödvändiga metoderna samt föreslå lämplig arbetsgång för utvecklingsarbete och kvalitetsarbete som innefattar införande av automatiska system för diagnos och övervakning.

Det kan vara intressant att försöka sammanfatta målet i en mening: *Att med vedertagna metoder på ett strukturerat sätt kunna påvisa för systemsäkerhet positiva såväl som negativa effekter efter införandet av diagnos- och övervakningssystem.*

1.3 Avgränsningar

Denna rapport kommer enbart att behandla så kallade irreparabla system. Dessa har karaktären att ett fel som uppkommer under drift inte

kan avhjälpas utan kvarstår under resterande driftsperiod. En konsekvens av detta blir att fokus för rapporten ligger på att analysera effekter med avseende på att undvika katastrofala händelser av engångskaraktär.

Rapporten syftar också till att framställa generella exempel snarare än att förvilla sig in i specifika typexempel. Detta leder till en något förenklad modell för sannolikhetsberäkningar, dock enligt vedertagna principer. Testsystemet som exemplifierar den framtagna metoden är ett gedankenexperiment. Därför existerar inte de data som behövs för en helt korrekt arbetsgång. De numeriska värdena som räknas fram är därför baserade på diskussioner med handledaren.

Av alla de metoder som används för säkerhets- och kvalitetsarbete har endast de som ansetts vara av centralt värde för rapportens mål tagits med.

1.4 Framtida arbete

Det är önskvärt att undersöka detta arbete med ordentliga indata från industrin eller åtminstone ett fullständigt projekt från tekniska högskolan. Med tillgång till ”äkta” projektdata skulle man kunna studera vilken effekt implementation av övervakningssystem haft på systemsäkerheten.

Det vore intressant att studera reparabla system utifrån underhållssynpunkt eller med tidsberoende sannolikheter. Den reparabla analysen är mer verkligehetstrogen och har fler intressanta analysresultat med underhållsrutiner som en av de viktigaste.

Ett annat konkret projekt är att forska kring hur en del sannolikheter bör beräknas. En hel del projekt är likartade i den bemärkelsen att operatörens situation är densamma. Med simuleringar torde man kunna ta fram statistiska data för olika reaktioner på olika stimuli.

Problemet med en tillförlitlig analys av mjukvara är även det ett område som skulle behöva studeras ur ett empiriskt perspektiv. Frågeställningen för detta lyder *Vilken analysmetod ska användas för mjukvara i ett specifikt system.* Vissa ledtrådar och pekpinnar finns i denna rapport.

Kapitel 2

Systemsäkerhet och kvalitetstekniska metoder

2.1 Tillförlitlighet samt diagnos- och övervakningssystem

Tillförlitlighet hos en process kan beskrivas som förmågan att göra rätt saker under en längre tidsperiod utan yttre ingripanden. För att öka tillförlitlighet i olika tekniska processer, har man länge använt sig av olika typer av övervakningssystem för att upptäcka avvikelser och reagera på lämpligt sätt. Automatiserad diagnos och övervakning av en process kan göras på en mängd olika sätt. Det mest traditionella konceptet diagnossystem utgörs av kontroll av gränsvärden för att upptäcka avvikelser i processen samt att i dessa fall slå larm. Ett typiskt exempel är en temperaturmätning där ett överskridet värde resulterar i att ett larm presenteras för en operatör. Det är sedan upp till operatören att agera. Man kan även tänka sig ett system där diagnosen automatiskt interagerar med styrningen av den övervakade processen.

Denna typ av diagnos är dock otillräcklig för många processer. Man har dels problemet med ett statiskt tröskelvärde som kan vara svårbedömt. Eftersom man önskar minimera antalet falsklarm, måste tröskelvärdet sättas till ett värde motsvarande ett värstafallsscenario, vilket resulterar i ett okänsligt system. Små fel samt indikationer på större fel förblir odetekterade. Andra problem är att en störsignal kan ge falsklarm och att felisolering är en omöjlighet. Detta problem med felisolering, att inte kunna identifiera den felande komponenten, är en svaghet i denna typen av system.

Tabell 2.1: Exempel på möjliga felmoder för en glödlampa.

Felmod	Beskrivning
F0	Felfritt
F1	Trasig glödtråd
F2	Trasigt glas
F3	Ingen ström

En annan traditionell metod för ökad tillförlitlighet är redundant hårdvara. Genom att ha dubbla, eller till och med trippla uppsättningar hårdvara, kan man uppnå god tillförlitlighet. Att ha redundanta system är dock dyrt, ökar vikten på produkten och kan antas lägga till ett visst mått av oönskad komplexitet till systemet [6, 9].

2.1.1 Modellbaserad diagnos och övervakning

Som ett alternativ till detta har man utvecklat modellbaserad diagnos- och övervakning [6]. Man skapar då en modell över processens funktion. Denna modell är en teoretisk uppskattning över hur processens olika delar fungerar och är oftast en rad matematiska samband som beskriver hur olika delsystem/komponenter uppför sig under körning. I modellen kan man även ta hänsyn till yttre störningar och därmed uppnå flexibla testgränser.

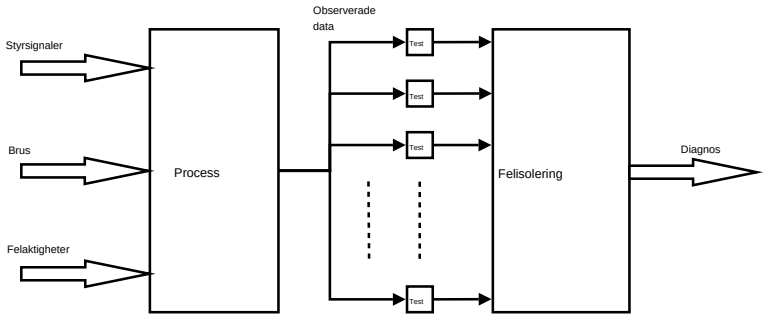
Man definierar sedan ett antal olika möjliga *felmoder* ett system kan befinna sig i. En felmod är en definition på hur väl systemet fungerar, inklusive fallet med ett felfritt system. Exempel på felmoder för en glödlampa skulle kunna vara som i tabell 2.1. Genom att sedan testa uppmätta processdata mot de definierade felmodernas karakteristik, var mod för sig, kan man teoretiskt sett genom uteslutning isolera fel med stor noggrannhet. Vinsterna med detta system är alltså många. Man kan upptäcka mindre fel, kompensera för störningar, få mindre reaktionstid och ha möjlighet till felisolering [6].

Diagnosen ställs utifrån den samlade kunskap som kommer av alla enskilda tester av felmoder. Diagnosen innehåller därför teoretiskt sett ett gott mått av felisolering. Resultatet kan också användas på två olika sätt.

1. Diagnossystemet presenterar diagnosen för en mänsklig operatör som tar beslut om processens fortsatta körning.

2. Diagnossystemet samverkar automatiskt med reglersystemet och dess fortsatta körning. Detta kallas *feltolerant styrning*.

Exempel på fall 1 är att larma visuellt samt med ljud. Exempel på fall 2 är om diagnossystemet automatiskt ändrar körningsmod för pro-



Figur 2.1: Principskiss över ett modellbaserat diagnossystem med hypotestest.

cessen. Exempel på detta kan vara att en upptäckt avvikelse leder till korrigerande åtgärder. Ett annat exempel är det som kallas feltolerant styrning, där man på ett genomgripande sätt ändrar systemets förutsättningar och körmod, till exempel ”limp home”-system för bilar eller flygplan. Ett exempel på ”limp home”-system för bilar, kan vara att om diagnossystemet upptäcker något fel, stryps maxfarten till 30 km/h så man åtminstone kan ta sig till en verkstad utan att riskera totalhaveri.

2.1.2 Reparabla och irreparabla system

Det finns två huvudsakliga systemtyper man måste skilja på. Med reparabla system menas system där ett fel kan uppträda i en komponent för att sedan försvinna. En tolkning är att felet kan avhjälpas under drift. Irreparabla system däremot behandlar fel som uppkommer under drift och inte kan avhjälpas; de kvarstår under resterande körtid. Som tidigare nämnts i avsnitt 1.3, kommer jag enbart att behandla irreparabla system. Detta då analyser på reparabla system behandlar en annan aspekt av produktkvalitet än den rena systemsäkerheten, nämligen tillgänglighet och underhållstäthet. Denna rapport fokuserar på processer för att undvika katastrofala engångshändelser - därav fokus på irreparabla system.

2.2 Systemsäkerhet

Systemsäkerhet är ett ord som sammanfattar en bred samling processer som bör implementeras i ett projekt för att uppnå maximal tillgänglighet och minimal felfrekvens. Objektet för projektet benämns *system*. Sveriges försvarsmakt har gett ut boken *H SystSäk E*[7], där man ger

sin version av hur projekt ska drivas ur systemsäkerhetssynpunkt. Denna boks definitioner av systemsäkerhet ligger som huvudsaklig grund för arbetet i denna rapport. I boken beskrivs de flesta processerna noggrant, men vissa ges väldigt lite plats. Till exempel beskrivs utförligt hur man arbetar med kvalitet på systemet i fråga. Detta beror på att metoderna för kvalitetsarbete är långt utvecklade och relativt integrerade i ingenjörstänkadet. Däremot talar boken om diagnos- och övervakningssystem med det korta beskedet att det ska finnas och att det ska implementeras så långt det är möjligt.

Vad man missar är hur man ska gå tillväga för att valda diagnossystem verkligen ska bidra till den totala systemsäkerheten. Att implementera dessa diagnossystem borde på något sätt anpassas till ingenjörsmässig standard. Processen ska vara integrerad med det övriga ingenjörsarbetet och metoderna som redan finns för kvalitetsarbete. Det finns en viktig poäng i att kunna påvisa de effekter ett diagnos- och övervakningssystem gör för total kvalitet och systemsäkerhet.

Två metoder har valts ut som intressanta och behandlas. *Failure Mode and Effect Analysis (FMEA)*, samt *Fault Tree Analysis (FTA)*. Orsakerna till att just dessa två valts ut är många och nedan listas de viktigaste[2, 9].

- Bägge är väl inkörda i industrin för kvalitetsarbete.
- Bägge ingår i standardiseringar av kvalitetsarbete och systemsäkerhetsarbete.
- De är de två mest centrala metoderna för att identifiera fel och felorsaker och det är rapportens kärna.
- I fallet FTA är det den metod som mest uppenbart kan användas för överskådligt arbete med kvantitativ analys av felrisker.

Allt som allt förefaller FMEA och FTA tillsammans utgöra en god grund för att kunna uppnå de uppsatta målen.

2.3 Introduktion till FMEA

Failure Mode and Effects Analysis (FMEA), är en dokumentdriven analysmetod för att behandla kvalitet på ett ingenjörsmässigt sätt. Den går ut på att analysera komponenter och delsystem med avseende på att finna felmoder och utreda dessa felmoders effekter på det totala systemet. Metoden utvecklades hos Bell och idag är FMEA väl utbrett inom en rad olika industrier för kvalitetsarbete. Förutom att FMEA är en ad hoc-standard inom industrin, ingår metoden även i ISO9001. FMEA utvecklades sedan vidare inom amerikansk flyg- och rymdindustri och finns i några olika varianter. Metoden fungerar enligt bottom-upprincipen på så sätt att man väljer ut enskilda komponenter i systemet

Tabell 2.2: FMECA-tabell.

Sannolikhet / **Allvarlighetsgrad**	Väldigt liten	Liten	Medium	Hög
Klass 1 - Små effekter				
Klass 2 - Signifikanta effekter				
Klass 3 - Kristiska effekter				
Klass 4 - Katastrofala effekter				

och bygger upp beskrivningar för deras potentiella inverkan på systemet om ett fel inträffar. FMEA är en dokumentdriven kvalitetsmetod, så det säger nästan sig själv att FMEA-formulären måste anpassas till det system man analyserar. Det finns alltså inget exakt standardformulär som alla använder. Dock följer alla samma generella mönster.

2.3.1 Process-FMEA

Process-FMEA behandlar fel på produkten som orsakas av störningar eller brister i tillverkningsprocessen. Resultaten kan sedan användas för att förbättra processen eller som ett underlag för hur man ska lägga upp arbetet med processtyrningen. Denna variant kommer därför inte att behandlas mer i denna rapporten.

2.3.2 Konstruktions-FMEA och FMECA

Konstruktions-FMEA är en analys av de ingående komponenternas felmoder och funktioner, både på lokal nivå såväl som på systemnivå. Detta är en kvalitativ analys, det vill säga att man finner komponenters svagheter och felpåverkan i systemet. Ibland kan man utöka denna metod med att söka en kvantifiering hos de felmoder som kan förekomma. I detta fall talar man om en *Failure Modes, Effects, and Criticality Analysis (FMECA)* [2, 9].

FMECA är en ”vanlig” FMEA utökad med en analys som bedömer sannolikheten och allvarlighetsgraden för varje felmod. I vissa fall bedöms även detekteringsförmågan, det vill säga sannolikheten att felmoden kan upptäckas *innan* den inträffat [2]. Det finns ett par olika sätt att åskådliggöra kvantifieringen. Antingen används en tabell med sannolikhet på ena axeln och allvarlighetsgrad på den andra [9], eller så talar man om *Risk Priority Number (RPN)* [2]. RPN definieras oftast som produkten mellan de tre talen felsannolikhet, allvarlighetsgrad och detekteringsförmågan, men även andra definitioner är möjliga [2]. Tabell 2.2 visar hur en en FMECA-tabell kan se ut.

Varje felmod som identifieras genom en FMEA, analyseras med avseende på kombinationen av felmodens frekvens och allvarlighetsgrad. Det krävs alltså en FMECA-tabell per felmod. FMECA-tabellen för en specifik felmod illustrerar då grafiskt hur kritiskt ett fel är. Ju längre ner mot höger hörn man kommer, desto allvarligare är felet.

2.3.3 Arbetsgång för FMEA

Generellt utgörs en FMEA av fyra huvudsteg [2, 9].

1. Dokumentation/definition av systemet, dess komponenter och funktioner.
2. Identifiering av komponenters och delsystems felmoder samt deras orsaker.
3. Analys av de identifierade felmoderna. Studera hur man detekterar dem och vad de har för effekter på systemet.
4. Slutsatser och rekommendationer.

Man får inte heller glömma att definiera för vilket operativt tillstånd systemet analyseras. Exempelvis har ett system som körs med maximal belastning under normala förhållanden helt andra förutsättningar än ett vilande system eller ett system som körs under extremt svåra förhållanden. Resultaten av en FMEA presenteras på en blankett som exemplifieras i figur 2.3. Ofta inkluderas även ett RPN, som redan på FMEA-stadiet ger en kvantiserad bedömning på hur stor risk komponenten utgör för systemsäkerheten. RPN kan senare vara en del i en algoritm för beräkning av felintensiteter och felsannolikheter i felträdsanalys.

2.3.4 Förutsättningar

För att kunna utföra en FMEA krävs mycket god kunskap om hur systemet fungerar som helhet, men även god kunskap om hur varje komponent fungerar och hur den påverkar systemet. Viktigast av allt vad gäller en FMEA är att man har tillräcklig kunskap för att kunna bedöma felbenägenheten. Vidare kan man omöjligt analysera alla beståndsdelar i komponenter och delsystem. Det skulle helt enkelt ta för lång tid och bli för komplext [9].

Nu framträder en annan viktig aspekt av FMEA - valet av vilka komponenter man ska analysera. Vissa delar måste tas med på grund av certifieringskrav, vissa har stor inverkan för den personliga säkerheten och andra är tydligt utpräglade felkällor som kan hota systemsäkerheten. Som ett exempel på urval kan nämnas att Sveriges försvarsmakt förespråkar att enbart nya eller ändrade komponenter måste analyseras [7].

Tabell 2.3: Exempel på FMEA-blankett.

FMEA-blankett									
Nr.	Komponent id	funktion	felmoder	felorsaker	Felets effekt på systemet	Upptäcks genom	Rekommenderad åtgärd	Ansvarig	Kommentarer
1.	P01_01 Bränslepump	Pumpa bränsle till motor	1.Fungerar inte	- Brusten drivaxel - Skurna lager	Motorstopp	Lågt flöde vid sensor S32_02	- Schemalägg kontroll av axel var tredje vecka - Använd enbart SKF kullager - Utred om ett automatiskt diagnossystem för lagerna kan minska felrisk.	JB	Beställning av diagnossystem for lagrena gjord.
			2.Nedsatt funktion	- Lager nära att skära	Dålig motor-prestanda.	Lågt flöde vid sensor S32_02	- Inför diagnos-system för lagerna	MK	

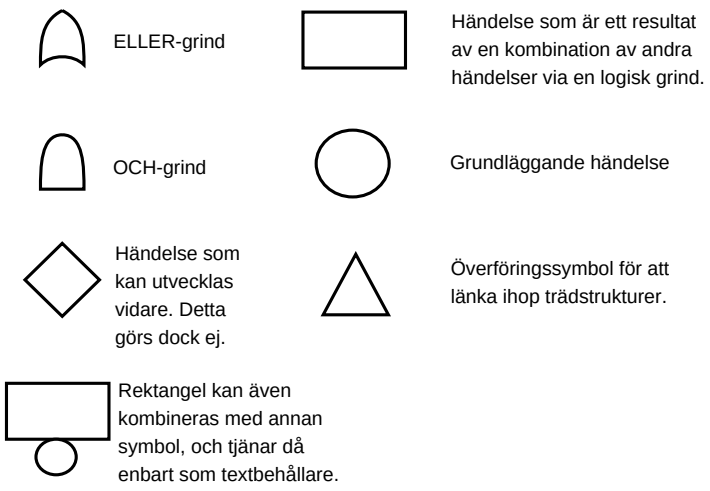
Det kan inte poängteras tillräckligt att en FMEA inte kan utföras på ett bra sätt om man inte är expert på systemet som analyseras.

2.3.5 Slutsatser från en FMEA

En FMEA kan ge en faktabas för många intressanta slutsatser. Bland de viktigaste bör nämnas följande.

- Identifikation av komponenter som kan ge upphov till allvarliga fel (FMECA).
- Identifiering av singulära felkällor. I områden som säkerhet för kärnkraft och rymdfarkoster är detta en primär funktionen.
- Inventering av systemets möjliga felmoder och deras möjliga inverkan på systemet.
- Kvantitativa kunskaper om fel och effekter (FMECA).
- Underlag för underhållsrutiner som matchar funna felmoder.
- Designunderlag för diagnostiska och övervakande system.
- Underlag för påföljande felträdsanalys.

Efter en genomförd FMEA kommer metoden med felträdsanalys som bygger vidare på resultaten, både på ett kvalitativt och ett kvantitativt sätt. FMEA kan i sig användas för att få fram samma resultat som en felträdsanalys, dock endast för komponenter vars fel direkt alstrar



Figur 2.2: De mest grundläggande elementen i ett felträd.

systemfel. För komplexa fel och större system krävs komplettering med felträdsanalys.

2.4 Felträdsanalys - FTA

FTA är en förkortning för *Fault Tree Analysis*, i vissa böcker även kallat *Cause Tree Analysis (CTA)*. Detta är en top-down metod, även kallat deduktiv metod, som används för att hitta svaga länkar i ett system och för att bedöma tillförlitlighet. Detta går även att göra direkt med FMEA, men enbart för komponenter som direkt leder till systemfel. Om det rör sig om samverkan av flera olika fel för att framkalla systemfel, är FTA ett nödvändigt redskap för analys.

Man gör detta genom att definiera en oönskad topphändelse och sedan arbeta sig ner nivå för nivå till de underliggande orsakerna till att händelsen inträffar. Varje underliggande händelse länkas in under toppnivån med hjälp av logiska grindar. På så sätt byggs ett logiskt träd upp som visar samband mellan enskilda händelser som leder till den oönskade topphändelsen. På den lägsta nivån av trädet kommer man att hitta enskilda komponenter och på så sätt se deras inverkan på systemet. På detta sätt kan man ändra kritisk design för att eliminera svaga länkar och singulära felkällor. I figur 2.2 visas de allra enklaste elementen som behövs för att bygga ett felträd. Det finns mängder med utökningar [9] som behandlar händelser med olika regler, tidsfördröjningar och mycket annat.

2.4.1 Kvalitativ FTA

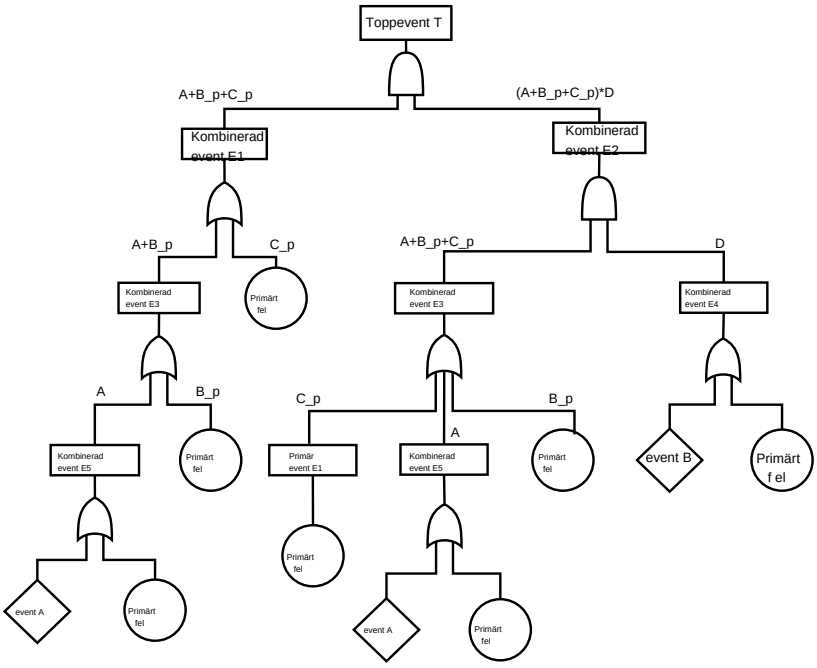
Målet med den kvalitativa delen av FTA är att hitta säkerhetskritiska länkar i systemet. Trädet analyseras för att ta fram väsentlig information om hur systemet måste byggas för maximal tillförlitlighet. Man vill främst eliminera singulära felkällor och i andra hand sekundära svaga länkar. Detta då de fortare leder till den katastrofala oönskade händelsen. En tumregel [9] är att enstaka komponenter inte får vara kritiska och att antalet OCH-grindar mellan komponent och topphändelse ger en snabb överblick av hur stor inverkan en komponent har på systemet.

2.4.2 Minimala cut sets

I större system blir dock felträd enormt stora och därmed oöverskådliga väldigt fort [9]. Istället för att analysera hela originalträdet, tar man fram något som kallas *minimala cut sets*. En cut set kallas ibland för *kritisk väg (critical path)*, och är de minimala sambanden av händelser som leder till den oönskade topphändelsen. Det vill säga att om en grundhändelse i ett cut set inte inträffar, så inträffar inte heller topphändelsen. En diagnostekniker har alltså ett utmärkt redskap i cut sets för att hitta vilka delar man bör prioritera för kvalitetshöjande åtgärder; till exempel övervakning med diagnossystem. Man vill finna så korta vägar som möjligt genom trädet eftersom dessa är mest kritiska för systemet. Notera att med korta vägar menas att så få händelser som möjligt måste inträffa samtidigt för att topphändelsen ska inträffa.

Det finns ett antal olika tekniker för att ta fram minimala cut sets. Den enklaste att beskriva är Boolesk algebra. Varje bashändelse tilldelas en Boolesk variabel och trädet analyseras sedan med hjälp av någon lämplig algoritm för att förkorta de Booleska uttrycken till disjunktiv normalform. Kvar blir då ett reducerat träd som beskriver de minimala händelseförloppen som leder till topphändelsen. Varje disjunkt konjunktion är då ett minimalt cut set. Det finns även en mängd andra algoritmer man kan använda. Eftersom felträd växer snabbt är det nödvändigt att använda sig av datorer för att göra analyserna. Det pågår ständigt ett sökande efter bättre algoritmer för att hitta cut sets så den intresserade föreslås leta bland artiklar hos IEEE eller ACM.

Figurerna 2.3 och 2.4 nedan illustrerar hur minimala cut sets fungerar. Trädet i figur 2.3 beskriver en oönskad topphändelse T. Varje händelse kan antingen vara ett singulärt fel, illustrerade med cirklar, eller kombinationer av flera typer av händelser, illustrerade med rektanglar. Dessa kombineras sedan med logiska grindar till nya kombinerade händelser. I figuren 2.3 finns sannolikheterna A, B_p, C_p och D utsatta. Grenarna under A och D har inga egna storheter eftersom de innehåller outvecklade händelser, istället tilldelas bara den kombinerade händelsen



Figur 2.3: Originalversion av felträd.

en sannolikhet. Genom att använda Boolesk algebra på grenarna kan topphändelsen skrivas som följer.

$$T = (A + B_p + C_p)(A + B_p + C_p)D$$

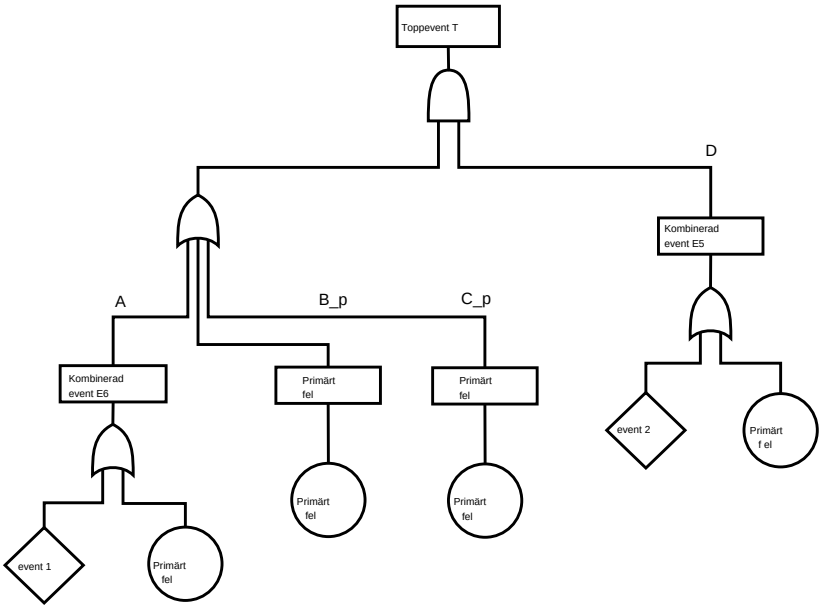
Som kan skrivas om med hjälp av lagen om idempotens ($X \cdot X = X$) till:

$$T = A \cdot D + B_p \cdot D + C_p \cdot D$$

Detta är det reducerade trädet som innehåller tre minimala cut sets:

$$A \cdot D, B_p \cdot D, C_p \cdot D$$

Nu ses en annan viktig detalj med minimala cut sets. Hade man resonerat kring felkällor på originalträdet, hade man lätt kunnat dra fel slutsatser. Man hade till exempel kunnat tro att topphändelsen T



Figur 2.4: Reducerat felträd.

beror på de tre händelserna som i figur 2.3 benämns $E3, E4$ och $E5$. I figur 2.4 över det reducerade trädet, kan man istället utläsa att T enbart beror på två händelser i taget - de ingående händelserna i de minimala cut sets som härleddes ovan.

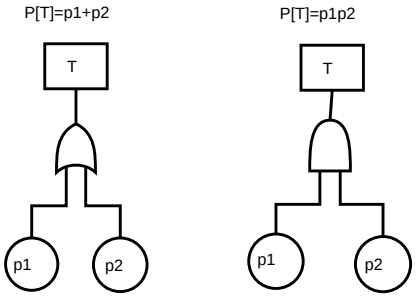
2.4.3 Kvantitativ FTA

Efter den kvalitativa delen av FTA kan man arbeta med den kvantitativa. Här letar man istället efter sätt att presentera numeriska storheter för systemets tillförlitlighet. Man vill alltså uttrycka sannolikheten att systemets oönskade topphändelser inträffar. För att kunna bedöma hur mycket en enskild komponent påverkar systemet, tilldelas varje komponent ett sannolikhetsmått. Detta mått beskriver komponentens förväntade felfrekvens.

2.4.4 Beräkning av sannolikhet för oönskad topphändelse

Det finns ett flertal metoder för att beräkna sannolikheten för en topphändelse. Dock visas enbart den enklaste eftersom metoden i sig inte har något värde för rapportens mål.

Här visas därför den enkla beräkningsmetod som uppkommer då



Figur 2.5: Kvantitativ beräkning av felsannolikhet enligt enkel direktmetod.

man antar oberoende mellan de ingående komponenterna [9]. Dessutom förenklas beräkningarna ytterligare av antagandet att de ingående felsannolikheterna är väldigt små och därmed kan ses som konstanta [9, 11]. Följande samband används [9].

- För ELLER-grindar enligt det vänstra trädet i figur 2.5 $P[T] \simeq P_1 + P_2$
- För OCH-grindar enligt det högra trädet i figur 2.5 $P[T] \simeq P_1 \cdot P_2$

Den intresserade läsaren kan hitta fler metoder, bland annat en baserad på minimala cut sets i den referade litteraturen [9] samt i en stor mängd tekniska rapporter.

2.4.5 Arbetsgång

Samma svårighet finns i FTA som i FMEA. Det är nödvändigt att man har en mycket god kunskap om systemet för att kunna göra en värdefull analys. Bästa sättet är enligt en rad författare inom ämnet att en systembeskrivning ska vara en del av analysen [1, 9]. Med denna kunskap ska man sedan definiera de topphändelser som är intressantast för en FTA. Det bästa sättet att skaffa sig god kunskap om systemet är att utföra en FMEA först. Det rekommenderas starkt att ha en arbetsgång med FMEA följt av FTA [2, 9]. Ett ytterligare plus med denna arbetsgången är nämligen den automatiska hjälp man får med det svåraste momentet i en FTA - att beräkna felsannolikheter. I en FMEA kan man då välja att räkna ut RPN - Risk Priority Number, för varje komponent. Detta RPN kan man sedan ha som stöd för beräkning av felsannolikheter [9]. En fullständig metod att räkna på felsannolikheter diskuteras i nästa stycke.

FTA kan användas på en rad olika sätt beroende av vad man önskar uppnå. Om målet är att eliminera designfel och öka kvaliteten i

allmänhet, lämpar sig en traditionell analys med felsannolikheter. Om man däremot vill minimera tiden som ett system är otillgängligt för drift, använder man sig lämpligast av en otillgänglighetsanalys [9]. En sådan innebär att man försöker hitta lämpliga intervall för underhåll av systemet. Andra exempel är analyser som inriktar sig mot medeltiden till första inträffade fel, överlevnadssannolikhet och försök att hitta svaga länkar.

2.5 Felsannolikheter och felintensiteter

Problemet att hitta tillräckligt bra värden på felsannolikheter och felintensiteter är ett svårt problem. Varje nod i ett felträd måste tilldelas ett värde inom godkända felramar för att analysen ska ge signifikanta resultat. Sannolikheten att en komponent inträder i en felmod är lika med ett då tiden går mot oändligheten. Därför antas att de delar som analyseras är nya och fungerar enligt specifikation. Felsannolikhet behandlas därför enligt tesen *fel under normal körning och normala förhållanden*. Det är nödvändigt att notera att detta inte är en exakt vetenskap. Just därför finns vissa pekpinnar som kan vara till hjälp. Största delen är dock baserat på erfarenhet och kunskaper om system och komponenter.

För varje enskild komponent kan man dock ta fram en tämligen god estimering av felsannolikhet. Den tidsberoende variabeln för felintensitet benämns vanligen λ. En metod som föreslås är att beräkna λ enligt följande modell [9].

$$\lambda = \lambda_b \pi_E \pi_A \pi_Q \pi_n$$

där de ingående värdena symboliserar

λ_b = Grundtes för felintesitet baserad på testdata och andra observationer etc.

π_E = En faktor som tar med den omgivande miljöns påverkan i beräkningar, temperatur exluderat. Olika miljö ger olika värden på π_E. Indelningen nedan är ett exempel för en godtycklig elektronisk komponent [9].

- 0,2: Normal omgivning på mark med optimala möjligheter för underhåll.
- 4 : Marknivå fäst vid mobil enhet $\Rightarrow$ utsatt för stötar och vibrationer.
- 10 : Svåra förhållande, uppskjuten med raket.

π_A = Operation adaptation factor; inkluderar specifika arbetspunkter som ger specifika felfall samt sekundär påverkan som kan influera beteende.

π_Q = Kvalitetsfaktor som ser till vilka design- och tillverkningsprocesser som använts [9], anger som exempel att en transistor kan ha ett π_Q mellan 0.2 och 10.

π_n = En faktor som ger möjlighet att justera för saker som antal repetitioner i en operationscykel etc.

Detta är en god översikt över de många faktorer som påverkar ett system. Saken är dock den att för analys över relativt korta tidsperioder och små sannolikheter, kan man slopa tidsberoendet och istället räkna statiskt med konstanta felsannolikheter [11]. Eftersom detta gagnar rapportens tydlighet och mål, kommer detta angreppssätt att användas.

Kapitel 3

Felbenägenhet hos diagnossystem

I detta kapitel diskuteras de olika element som bygger upp ett diagnossystem och vad som krävs för att kunna analysera dem kvantitativt med avseende på felbenägenhet. Störst fokus läggs på analys av mjukvara, där en kombination av ett flertal analysmetoder förespråkas.

3.1 Vad bygger upp ett diagnos- och övervakningssystem?

Införande av ett diagnossystem leder alltså till att en rad nya komponenter och delsystem läggs till det totala systemet. Det måste finnas fyra grundläggande saker.

1. Sensorer/mätinstrument som kan fånga upp de önskade signalerna samt medium för överföring av dessa signaler.

2. Datorarkitektur (minne, buss, processorkraft etc.) för att köra diagnosrutinerna.

3. Interface, det vill säga möjlighet att automatiskt interagera med huvudprocessen eller möjlighet att presentera diagnosen för mänsklig mottagare.

4. Programkod för diagnossystemet.

3.2 Bedömning av felbenägenhet i sensorer och övrig hårdvara

Sensorer, kablage och andra hårdvarukomponenter som införs i systemet kan potentiellt fallera på två olika sätt. Antingen kan den uppmätta signalen förvanskas eller så slutar komponenterna helt att fungera. I bägge fallen kommer systemet att agera på ett sådant sätt att en felaktig diagnos blir resultatet. En lindrig variant av felaktig diagnos är antagandet att komponenten i fråga slutat fungera och därmed slutar diagnoserna användas; betydligt allvarligare är fall då diagnosen triggar ett felaktigt automatiskt körmodsbyte [6].

Denna typ av komponenter har dock fördelen att de oftast är väl dokumenterade. Elektriska komponenters prestanda finns tabellerade, antingen av tillverkaren själv eller i de stora MIL-standard-tabellerna. Via denna kunskap kan man använda metoden som beskrevs i avsnitt 2.5 för att arbeta fram bra värden för felsannolikheter.

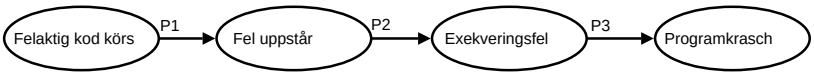
3.3 Bedömning av felbenägenhet i datorarkitektur

Datorkraften ger andra biverkningar. Påförandet av ett stort diagnossystem kan eventuellt bidra till att en kraftfullare dator måste användas. Med kraftfullare menas större minne, större buss och framförallt en kraftigare och mer avancerad processor. Generellt sett är det så att äldre processorers prestanda och inbyggda svagheter, är väl dokumenterade, så man kan programmera sig runt eventuella problem. Om man tvingas byta till en nyare processor finns risken att en del nya fel uppkommer. Dessa fel är inte kända till sin natur och kan potentiellt vara förödande. Industrin drar sig därför in i det längsta för att byta fungerande datastrukturer om det inte verkligen är nödvändigt [3].

Bedömningen ligger i viss mån hos hårdvaran, då datorarkitekturen består av de komponenter som avses i avsnitt 3.3. Men hänsyn måste även tas till hur beprövad ny teknologi är. Orsaken till oviljan att byta processorer är den ökade felbenägenheten hos en ny teknologi [3].

3.4 Bedömning av fel i diagnospresentation

Presentationsmedium för diagnosen är förmodligen ett mindre problem. Det handlar i de flesta fall om att presentera diagnosen för en mänsklig operatör. Dessa felaktiga diagnoser är i samtliga fall felaktiga larm. Risken i dessa fall ligger i att operatören kan ta ett felaktigt beslut med falsklarmet som faktabas. I det andra fallet där diagossys-



Figur 3.1: Att ett stycke felaktig kod körs innebär inte automatiskt att felet får konsekvenser. Följande kedja måste vara komplett: p_1 = felaktig kod körs, p_2 = exekveringsfel uppstår och p_3 = exekveringsfel → programkrasch.

temet automatiskt ska byta körmod för den övervakade processen, kan konsekvenserna bli betydligt värre.

Bedömning av dessa kriterier är desto svårare. För analys av mänskliga fel rekommenderas metoder som *Systematic Human Action Reliability Procedure (SHARP)* och *Tecnica Empirica Stima Errori Operatori (TESEO)* [10].

3.5 Analys av felrisk i mjukvarukod

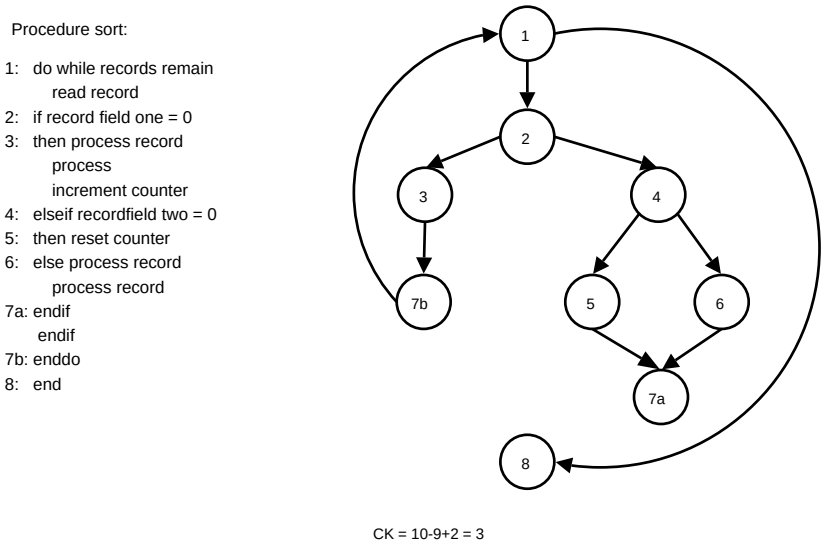
En önskedröm här vore att med någon sorts algoritm kunna bedöma hur felbenägen kod är. Tyvärr finns ingen generell metod att utifrån källkod estimera felsannolikhet. Dessutom är det inte säkert att ett fel i programkoden leder till varken exekveringsfel eller programkrasch. Om just den felaktiga biten kod sällan körs, eller körs utan att alstra exekveringsfel, så händer inget. Om koden körs med exekveringsfel som följd, behöver inte detta leda till programkrasch. Figur 3.1 illustrerar sannolikheten för att ett allvarligt programfel ska uppstå. Det finns tre huvudsakliga sätt att undersöka och bedöma kod utifrån statiska data [10].

1. Strukturell komplexitet
2. Datakomplexitet/Kodmassa
3. Användningsgrad och kommunikation med andra moduler.

Utöver dessa finns även metoder som bedömer kod empiriskt - fault seeding och fault injection [4].

3.5.1 Cyklomatisk komplexitet - McCabes teorem

År 1976 introducerade Thomas McCabe en metod för att bedöma strukturell komplexitet i ett program. Begreppet kallas *cyklomatisk komplex-*



Figur 3.2: Exempel på analys enligt McCabes cyklomatiska komplexitetsteori.

itet (CK) [10]. Metoden går ut på att skapa ett flödesschema över koden, se exemplet i figur 3.2. Varje oberoende väg skapar en förgrening i grafen och den cyklomatiska komplexiteten kan bland annat beräknas som

$$CK = E - N + 2$$

där

$$E = \text{antalet bågar i grafen}$$

och

$$N = \text{antal noder i grafen}$$

Talet representerar ett mått på hur komplex struktur ett program har. Tester har visat att korrelation föreligger mellan den cyklomatiska komplexiteten och felbenägenhet hos program. Dock finns tester som påstår det rakt motsatta [8]. Metoden ger dock ett komplexitetsmått på ett tidigt stadium i utvecklingen av ett program. Man ska vara medveten om de svagheter metoden besitter. Cyklomatisk komplexitet lägger ingen vikt på datakomplexitet utan bara på programflödeskomplexitet. Detta gör att enkel aritmetik klassas lika hårt som komplicerade beslutsstrukturer.

3.5.2 Halsteads metod, 4-tupelmetoden

En annan variant på kodanalys är Halsteads metod som istället fokuserar på datakomplexitet [8]. Det skall dock direkt nämnas att Halsteads

metod enbart kan användas på fullständigt känd kod då den fokuserar på att räkna element i koden. De räkningsbara delarna är

- n_1 = antalet unika operatorer i programkoden
- n_2 = antalet unika operander i programkoden
- N_1 = total användning av kodens alla operatorer
- N_2 = total användning av kodens alla operander.

Med operatorer menas addition, multiplikation och andra operatorer man väljer att ha med i sin algebra. Med operander menas de objekt operatorerna verkar på. Exempel på operator kan vara addition, och operander kan vara två variabler x och y som tilldelas värden och adderas. Med dessa nyckeltal definierar Halstead den *mentala arbetsinsatsen (M)* för att skapa ett program, en numerisk storhet som han även verifierade med experiment och fann korrelationer till antalet fel i ett program. Man kan notera att antalet buggar N, är i storleksordningen

$$N = \frac{M^{2/3}}{3200}$$

Själva storheten V har följande definition.

1. En vokabulär $n = n_1 + n_2$
2. En implementationslängd $N = N_1 + N_2$
3. Kodvolym $V = Nlog_2n$ gäller som storleksmått på antal bitar som behövs för implementationen.
4. Programnivå $L = (2/n_1)(n_2/N_2)$ är ett mått på vilken nivå programmet kan förstås.
5. Programmets svårighetsgrad $D = 1/L$
6. Slutligen definitionen av M: $M = D * V$.

Halstead är vida accepterat inom industrin och används idag främst för att räkna på underhållsnivåer och testbehov. Samma problem som för McCabes metod finns dock. Vissa tester anser att det går att påvisa felbenägenhet medan andra anser precis tvärtemot [8].

3.5.3 KLOC

KLOC står för kLines Of Code, det vill säga antal rader kod i tusental. Måttet är alltså ett rakt storleksmått på programlängd. Tester har visat vissa korrelationer mellan KLOC och antal fel i koden. Återigen finns dock undersökningar som pekar på det motsatta. Generellt mått antas vara ett fel per KLOC.

3.5.4 Fault seeding och fault injection

Fault seeding går ut på att plantera ett antal fel i programvaran [4]. Dessa fel antas vara likvärda de existerande felen i programmet vad gäller svårighet att upptäcka samt allvarlighetsgrad. Vid tester kan man sedan jämföra hur många av dessa planterade fel som upptäcks mot det totala antalet planterade fel och antalet funna ”riktiga” fel. Med hjälp av dessa data kan det totala antalet ”riktiga” fel beräknas.

Fault injection är en sorts vidareutveckling av fault seeding. Grundidén bakom metoden är att man hanterar *effekterna* av eventuella fel, snarare än att försöka rätta alla buggar. Metoden fungerar på så sätt att man i en simuleringsmiljö ”sätter krokben” för programmet. Detta kan göras genom att ändra, lägga till, eller ta bort kod. Man studerar effekterna och skapar på så sätt en bedömning över hur programmet kommer att bete sig i händelse av eventuella fel. Denna data kan sedan användas för att förbättra programvarans robusthet.

3.5.5 Slutsatser om kodanalys

De metoder som beskrivits ovan har troligtvis liten nytta var för sig. Kombinerade kan de dock ge fingervisningar om hur programkodskvalitet kan uppskattas. McCabe ger ett komplexitetsmått baserat på flöden, Halstead studerar datakomponenter i koden, KLOC ser till mängd, fault seeding och fault injection är empiriska metoder för bedömning. Mängden kommunikation som sker med andra datainstanser är även den ett viktigt bedömningskriterium. Följande parametrar måste sannolikt även beaktas i eventuella beräkningar [8, 9] .

- Programspråket
- Utvecklingsprocessen och utvecklarnas kunskap (Kan även tolkas in i Halsteads begrepp *mental arbetsinsats*)
- Kompilator
- Målsystem
- Sannolikheten att koden körs och alstrar fel.

Totalt sett borde det gå att samla data för hur program uppför sig i vissa system och på så sätt uppskatta dess felbenägenhet med statistik.

Kapitel 4

Utökning av systemsäkerhetsarbete för diagnos- och övervakningssystem

Kan man använda sig av de metoder som beskrivits i kapitel 2 respektive 3 för att på ett strukturerat sätt påvisa de systemsäkerhetsmässiga effekter som uppkommer då ett diagnossystem införs i ett system? Mot bakgrund av målformuleringen i avsnitt 1.2 formuleras en kravspecifikation. En arbetsmetod beskrivs och testas sedan med data från ett befintligt projekt som drivs vid Linköpings tekniska högskola [5].

4.1 Antagande om utgångspunkt

För att arbetet med att analysera diagnossystem ska vara meningsfullt, krävs vissa förutsättningar och förenklingar. I verkligheten är diagnossystemen sammansmälta med det totala systemet i designfasen såväl som i utvecklingsfasen. Detta gör en *före-efter*-jämförelse svårartad mellan ett system med, respektive utan diagnossystem. Det förutsätts därför att utgångspunkten i arbetet är att ett system har analyserats för brister och ett diagnossystem har föreslagits som åtgärd för att eliminera svaga länkar och punkter med hög felbenägenhet. Nu önskas ett systematiskt förfarande för arbetet med diagnossystemet med målet att skillnader före-efter ska kunna åskådliggöras.

4.2 Kravspecifikation

För att skapa en arbetsgång har en kravlista tagits fram med hjälp av litteraturstudier och funderingar med handledaren. Resultatet är fyra krav som anses vara lämpliga för att kunna skapa en användbar men ändå generisk metod.

4.2.1 Krav 1 - följ standarder

Kravet adresserar användbarheten hos en metod utifrån användarnas acceptans. Ett sätt är att utnyttja existerande metoder som av användaren uppfattas som beprövade och effektiva. Det skapar trovärdighet och legitimerar resultaten. Dessutom syftar kravet till att säkra effektiviteten rent praktiskt, det vill säga att metoden faktiskt uppfyller den funktion man önskar använda den för. En standardiserad metod har undersökts av standardiseringsinstitut och industrin och via dessa bevisats effektiv. På så sätt är man försäkrad att faktiskt nå de resultat man önskar.

Kravet har varit en undermedveten röd tråd i hela rapporten. Därför ligger fokus på FMEA och FTA som centrala metoder då dessa är lämpligast för ändamålet.

4.2.2 Krav 2 - generell metod

Detta krav tar hänsyn till att systemsäkerhetsarbete med FMEA och FTA varierar med systemtypen. Därför är det önskvärt att definiera en generell metod, som är användbar för så många olika systemtyper som möjligt. Detta skapar mervärde för metoden som annars hade varit väldigt begränsad i användbarhet.

4.2.3 Krav 3 - återspegla effekterna tydligt

Här berörs de delsystem som i tidigare analyser visats behöva övervakning av ytterligare diagnossystem. Efter att ett sådant diagnossystem införts, ska de positiva effekterna på de berörda delssystemen tydligt framgå och kunna förstås. Detta innebär att metoden ska beskriva hur den kvantitativa analysen utan en mängd merarbete ska kunna modifieras med nya värden för felsannolikhet.

4.2.4 Krav 4 - hantera diagnossystemet på ett enkelt sätt

Kravet syftar till att ge ingenjören en överblickbarhet och en känsla för de bieffekter som diagnossystemet orsakar. Diagnossystemet ska analyseras med avseende på de negativa effekterna som eventuellt uppstår. I

detta fall önskar man därför kunna arbeta med diagnossystemet som ett homogent system. I ett felträd innebär det praktiskt att man vill hantera diagnossystemet som en enda gren, eller åtminstone som så få grenar som möjligt.

4.3 FMEA som kvalitativ underlagsanalys

För att uppfylla kraven inleds analysen med en ny FMEA. Som beskrivits i avsnitt 2.3, finns ingen standard-FMEA. Följande punkter är dock krav i den FMEA som ska genomföras.

1. Minst varje delsystem/komponent som påverkas av diagnossystemet analyseras på nytt. Notera att detta följer rekommendationer i H SystSäk E [7].

2. Eventuella komponenter som läggs till för att introducera diagnossystemet analyseras också. Detta måste även omfatta program/programkod. Även detta enligt H SystSäk E [7].

3. För bägge ovanstående punkter ska effekterna av diagnossystemen noteras med en verbal beskriving samt ett RPN.

I delarna som skrivits om FMEA i avsnitt 2.3, nämns vikten av mycket god systemkännedom. Denna poängteras återigen. Vikten av en väl genomförd FMEA med god systemkännedom är kritisk i konstruktionen av felträd.

4.4 FTA som kvantitativ analysmetod

Nu följer en felträdsanalys för diagnossystemet. Främst i kvantitativt syfte. Med minimala cut sets enligt avsnitt 2.4.2 kan även en kvalitativ analys genomföras. Fokus här ligger dock på den kvantitativa biten som beskrivs i avsnitt 2.4.3.

Först analyseras de delar som påverkas av diagnossystemet, det vill säga de delar som övervakas. Det finns två principsätt att göra detta på.

1. Det enklaste sättet är att för varje påverkad händelse på nytt beräkna felsannolikhet med hänsyn taget till diagnossystemets förbättrande effekter. Det nya värdet används sedan för händelsen i felträdet.

2. Kombinera den övervakade händelsen med en *diagnoshändelse* via en OCH-grind. Diagnoshändelsen ges en sannolikhet motsvarande hur stor procent av den övervakade händelsens fel som släpps igenom.

Fördelen med variant ett är att man slipper rita om trädet. Dock finns ingen representation för att sannolikheten för händelsen reducerats genom ett yttre system. Ingenjören kan då luras att tro att komponenten/delsystemet är säkrare i sig självt än vad som faktiskt är fallet. Vid eventuella omräkningar av felsannolikheten hos en komponent, måste även den ursprungliga felsannolikheten finnas tillgänglig för korrekta beräkningar. Det krävs alltså att två värden lagras - ursprunglig sannolikhet och diagnossystemets inverkan.

Fördelen med variant två är den grafiska representationen av yttre påverkan. Nya beräkningar sker dessutom genom den logiska grinden, genom att enbart ändra händelsernas sannolikhetsvärden var för sig. Nackdelen är att trädet måste ritas om vilket kan vara ett stort projekt då det kan röra sig om hundratals ändringar på olika ställen i trädet.

Vilken av dessa två metoder man väljer är oväsentligt då de ger samma resultat. Om det inte är absolut nödvändigt med en tydlig graf, rekommenderas den förra metoden över den senare, just med arbetsmängden som argument.

Nästa steg är sedan att analysera den negativa inverkan diagnossystemet har på det totala systemet.

1. Gör en lista över alla oönskade topphändelser som diagnossystemet kan leda till och som samtidigt är oönskad händelse i analysen av det ursprungliga systemet. Om ingen sådan topphändelse existerar, kan man utesluta diagnossystemet som felkälla och avsluta analysen.

2. Skapa ett nytt felträd. Topphändelsen i det nya trädet är en av de oönskade händelserna i punkt 1.

3. Konstruera ett felträd under den valda topphändelsen i punkt 2 enligt reglerna för felträdskonstruktion.

4. Om det existerar kända sannolikheter för falsklarm, ska dessa länkas in i felträdet som egna händelser.

5. Beräkna felsannolikheter för alla händelser i felträdet. Beräkna sedan sannolikheten att trädets topphändelse ska inträffa.

6. Lägg till diagnosträdets topphändelse med en ELLER-grind direkt under motsvarande händelse i det ursprungliga systemets felträd.

7. Gör om punkterna 2-6 för hela listan i punkt 1.

Nu kan man återigen analysera sannolikheten för topphändelsen och få ett mått på diagnossystemets effekter.

4.5 Varför denna modell för arbetsgång?

En berättigad fråga är varför just denna arbetsmodell föreslås. Svaret är att detta är en bedömning baserad på litteratur, diskussioner och tester. Ett sätt att motivera valet av modell är att diskutera igenom varje krav mot metoden.

4.5.1 Krav 1 - följ standarder

Valet av FMEA och FTA som de centrala metoderna är gjort delvis på basis av det genomslag de haft i industrin. Bägge metoderna ingår som tidigare nämnts i en rad standarder [7, 9]. FMEA och FTA är dessutom etablerade i existerande programvara för kvalitetsarbete. Krav på själva användningen av metoderna, som att enbart de påverkade komponenterna ska analyseras efter påfört diagnossystem, påpekas i försvarets beskrivning av systemsäkerhet [7]. Sammantaget torde det på det hela taget vara enklare att få genomslag för denna typ av analys med de bekanta metoderna FMEA och FTA.

4.5.2 Krav 2 - generell metod

Metoden är beskriven i generella ordalag utom vid vissa delar där det krävs tydligare regler för att undvika förvirring. Speciellt noga är att inga systemspecifika beskrivningar finns medtagna. Det poängteras i avsnitt 4.3 att en god systemkännedom krävs. Metoden i sig är inte självgående utan kräver en kunnig utövare. Arbetsgången för FTA i avsnitt 4.4 är speciellt framtagen med generalitet i åtanke. Det anges två olika sätt att analysera de övervakade delsystemen, just för att kunna ha alternativ vid olika behov. Andra typer av specificeringar har undvikits, till exempel nämns inte hur en felsannolikhet ska beräknas, utan det hänvisas endast till de riktlinjer som finns beskrivna i kapitel 2 och 3.

4.5.3 Krav 3 - återspegla effekterna tydligt

I avsnitt 2.4 framställs fördelarna med felträdsanalys som ett redskap för att analysera effekter, kvalitativt såväl som kvantitativt. Det något förenklade systemet att använda statiska felsannolikheter stöds i andra vetenskapliga rapporter [11] och beskrivs som adekvat i avsnitt 2.5. Förenklingen att endast analysera utifrån irreparabla system är en vald avgränsning som beskrivs i avsnitten 1.3 och 2.1.2. FTA anses ge de bästa resultaten och den enklaste arbetsgången.

4.5.4 Krav 4 - hantera diagnossystemet på ett enkelt sätt

Detta krav gäller främst FTA. I avsnitt 4.4 beskrivs arbetsgången för FTA. Punkt 1 gör där gällande att första steget är en listning av topphändelser som även existerar som händelser för det totala systemet. Finns inga sådana har diagnossystemet inga intressanta negativa effekter. Vid existens av någon dylik händelse, analyseras denna som ett eget system och länkas enkelt in i det ursprungliga systemets felträd. Analysen är på så vis enkel och rättfram.

4.6 Systemsäkerhetsprojekt med bilbana som principsystem

För att demonstrera den metod som nu beskrivits i föregående avsnitt 4, visas i detta avsnitt ett exempel baserat på ett befintligt projekt. Projektet är ett gedankenexperiment, vilket specifikt innebär att händelsernas felsannolikheter är baserade på resonemang inom gruppen snarare än regelrätt analys.

4.6.1 Inledning

Vid avdelningen för fordonssystem finns en automatreglerad bilbana. Denna har använts som principsystem vid ett projekt som syftade till att skaffa djupare förståelse för systemsäkerhet, dess metoder och sambandet mellan systemsäkerhet och diagnossystem [5]. Dock gjordes analysen enbart på diagnossystemets positiva effekter. De negativa effekterna som diagnossystemet i sig eventuellt kan ha på systemet, beaktades alltså inte. Syftet med detta exempel är att nu även analysera de negativa effekterna. Först beskrivs därför det ursprungliga projektet i avsnitt 4.6.2 till och med avsnitt 4.6.4. Sedan utökas analysen i avsnitt 4.6.5 till och med avsnitt 4.6.7 med den metod som beskrivits tidigare i kapitlet.

Utgångspunkten för experimentet är behovet för flygplanskonstruktörer att kunna få flygtillstånd för sina flygplan. Detta utfärdas efter att planet bedömts som *flygvärdigt*. Flygvärdighet bedöms som sannolikheten för plankrasch per flygtimme. För principsystemet med en datorstyrd bilbana, finns möjligheten att identifiera liknande parametrar som hos ett flygplan.

4.6.2 Principiella likheter mellan bilbana och flygplan

I undersökningen mappades de vitala parametrarna från flygplansfallet ner på motsvarande parametrar för bilbanan. Topphändelsen för flygplanet, antal krascher per flygtimme, motsvaras i bilbanan av antalet gånger bilen åker av banan per körtimme. Eftersom analysen beror av ett antal miljöfaktorer, det vill säga var och hur systemet används, har även sådana parametrar betraktats i avsnitt 2.5. Om man till exempel avser att låta bilen stå stilla på banan, blir risken för avåkning noll. Därför finns krav på referenstid, det vill säga hastighetskrav, inlagda för att motsvara flygplanets uppdragsprofil och uppdragstid. Andra parametrar är styr- och reglermjukvara och hårdvara samt den mänskliga operatörsfaktorn.

4.6.3 Analys av bilbana utan diagnos

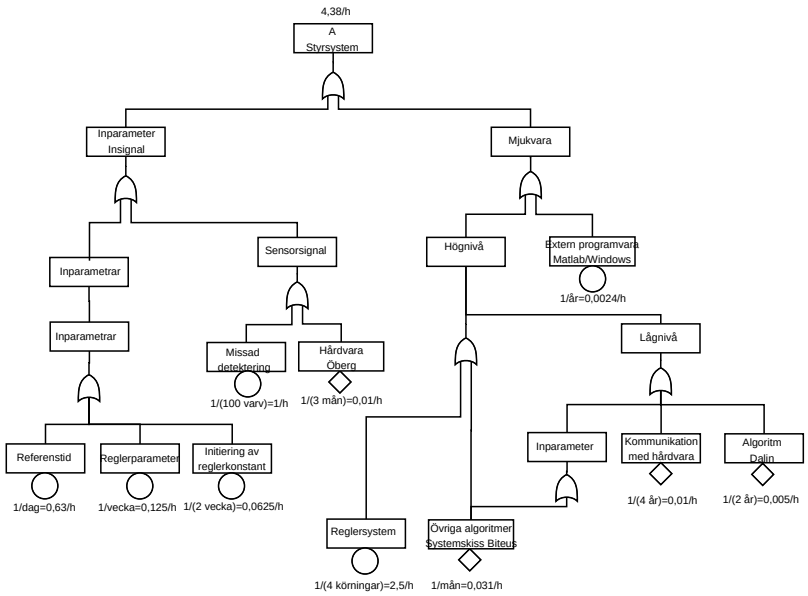
Det ursprungliga systemet utan något diagnossystem inkopplat analyserades först, bland annat med FMEA och FTA. För överskådlighetens skull delades felträdet upp i två delar. I figur 4.1 visas felträdet över styrsystemet och i figur 4.2 visas hela bilbanesystemet. Notera att styrsystemet är inlänkat direkt under toppnivån längst till vänster i det totala systemet i figur 4.2. Antal avåkningar totalt sett har beräknats till 5,08 per timme. Majoriteten av dessa kan härledas från styrsystemet som bidrar med 4,38 avåkningar per timme. Beräkningarna av felintensiteter är baserade på att man kör 100 varv per timme, åtta timmar per vecka, 5 dagar per vecka, 4 veckor på en månad och 52 veckor på ett år.

4.6.4 Analys av bilbana med diagnos

I projektet gjordes sedan en analys där effekterna av ett diagnossystem har tagits med. Dock enbart de positiva effekterna då denna rapport ännu inte skrivits. I figur 4.3 visas styrsystemet, nu med tre olika fall av diagnossystem tillagt.

1. Parametern *referenstid*, längst ner till vänster i figur 4.3 har via en OCH-grind kombinerats med *övervakning ref. tid* som enligt figuren motsvarar ett diagnossystem som hanterar 50% av de felaktiga fallen.

2. Parametern *missad detektering*, längst ner i bild ungefär i mitten av figur 4.3, har kombinerats med *FDI-system* som enligt figuren motsvarar ett diagnossystem som hanterar 95% av de felaktiga fallen. I figuren används en OCH-grind för att länka ihop diagnossystemet med det övervakade systemet. Därför används 5%



Figur 4.1: Ursprunglig felträdsanalys över bilbanans styrsystem.

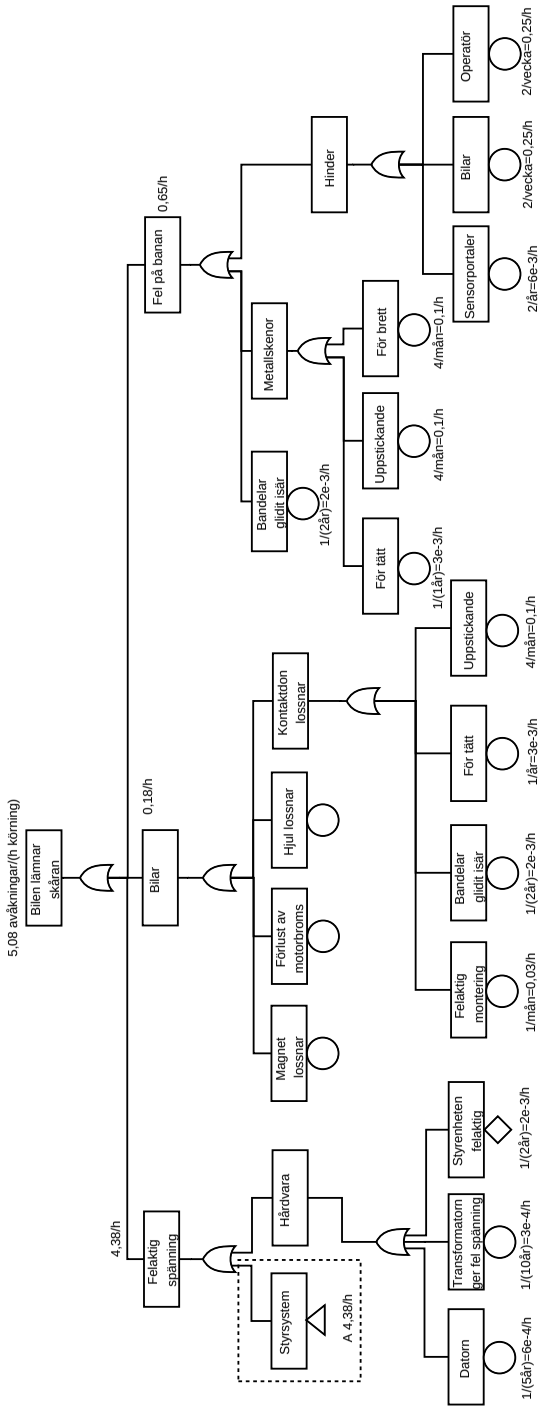
som beteckning, eftersom diagnossystemet via OCH-grinden släpper igenom 5% av de felaktiga fallen. FDI står för *Fault Detection and Isolation*.

3. Parametern *reglersystem*, längst ner i bild strax till höger om föregående parameter, har kombinerats med *lågnivå begr. pådrag*, som enligt figuren motsvarar ett diagnossystem som hanterar 75% av de felaktiga fallen. Talet 25% används av samma anledning som beskrivs i punkten ovan.

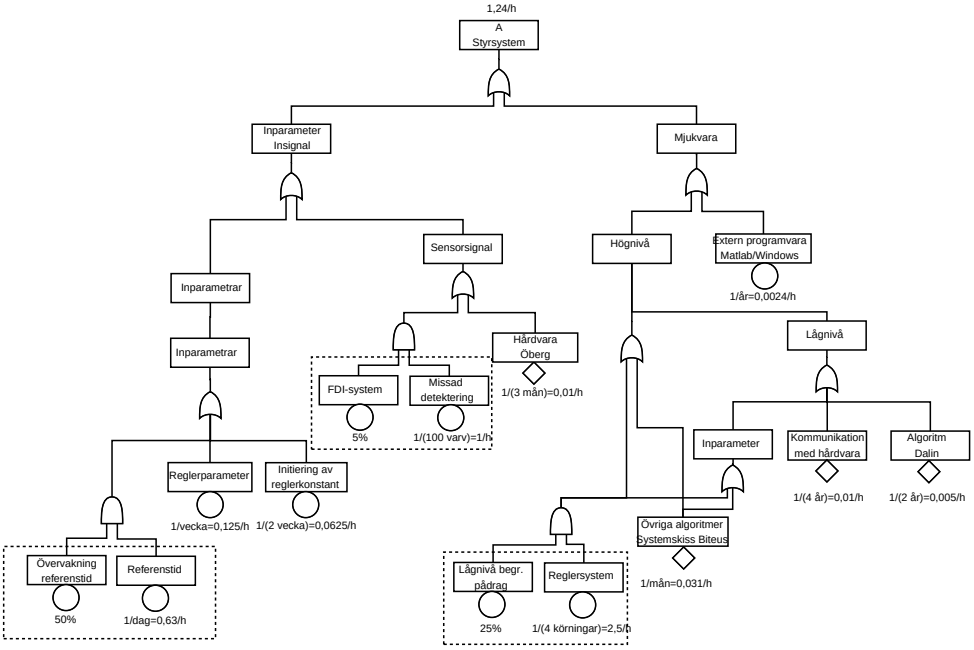
Med diagnossystem blir bidraget från styrsystemet 1,24 avåkningar per timme. Figuren 4.4 visar det totala systemet med diagnossystem. Antalet avåkningar per timme har reducerats till 2,07 per timme.

4.6.5 Utökad analys av diagnossystemets effekter

Som nämndes i avsnitt 4.6.4, analyserades aldrig några negativa effekter som diagnossystemet skulle kunna tillföra. Diagnossystemet ska därför nu analyseras enligt den metod som presenterades i början av detta kapitel. Först måste det tillförda diagnossystemet delas upp i sina beståndsdelar enligt avsnitt 3.1. Detta för att kunna utföra en analys överhuvudtaget. Tabell 4.1 visar de tre modulerna som enligt avsnitt 4.6.4 bygger upp diagnossystemet, samt identifierar deras beståndsdelar.



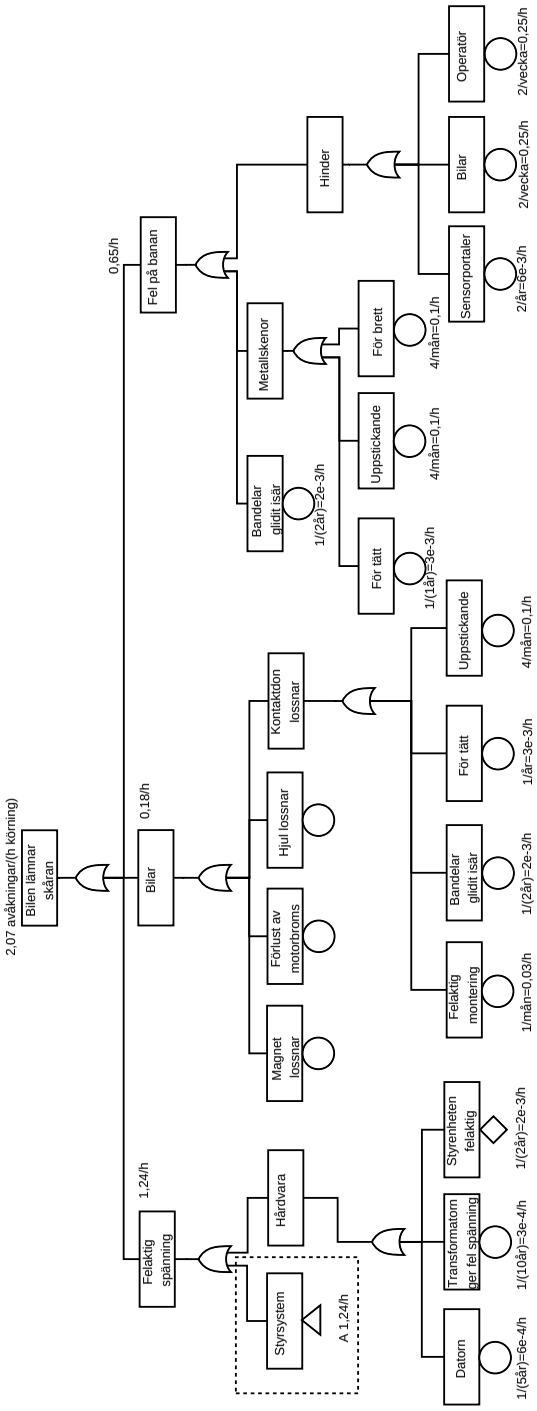
Figur 4.2: Ursprunglig felträdsanalys över bilbana. Styrsystemet är inlänkat längst till vänster och är inramat.



Figur 4.3: Felträdsanalys över bilbanans styrsystem efter utökning med diagnossystem. De delar som belagts med diagnos har ramats in i figuren.

Tabell 4.1: Tabell över de delar som diagnossystemet tillfört.

Egenskaper / **Tillfört**	Består av
Övervakning av ref. tid	programkod
FDI-system	programkod
Lågnivå beg. pådrag	programkod



Figur 4.4: Felträdsanalys över bilbana med diagnossystem. Sannolikheten för att styrsystemet ska ställa till det har minskar avsevärt och sannolikheten för att topphändelsen ska inträffa är minskad till 2,07 från originalsystemets 5,08.

Tabell 4.2: FMEA över bilbanans utökning med diagnossystem.

FMEA diagnossystem									
Nr.	Komponent	funktion	felmoder	felorsaker	Felets effekt på systemet	Frekvens	Allvarlighetsgrad	RPN	Kommentarer
1.	Övervakning av referenstid	Kontroll att inmatad ref. tid är inom giltiga värden.	Referenstid utanför giltiga värden.	- Programfel - Modellbrister	- Hög friktion kan göra att bilen står still med missade checkpoints som resultat.	F	2	12	JB ansvarig utvecklare
					- För högt gaspådrag.	F	2	12	JB ansvarig utvecklare
2.	FDI-system	Övervakning av missade checkpoints	Felaktig bedömning av missad checkpoint	- Modellbrister - Programfel - Sensorfel	- Felaktigt gaspådrag för att kompensera för felbedömning	D	2	8	
						F	2	12	
3.	Lågnivå begr. pådrag	Begränsar max gaspådrag	Felaktigt gaspådrag	- Modellbrister	- Modellbrister ger felaktigt gaspådrag	D	2	8	
				- programfel	- Fel i program ger felaktigt gaspådrag	F	2	12	

4.6.6 FMEA över diagnossystemets negativa effekter

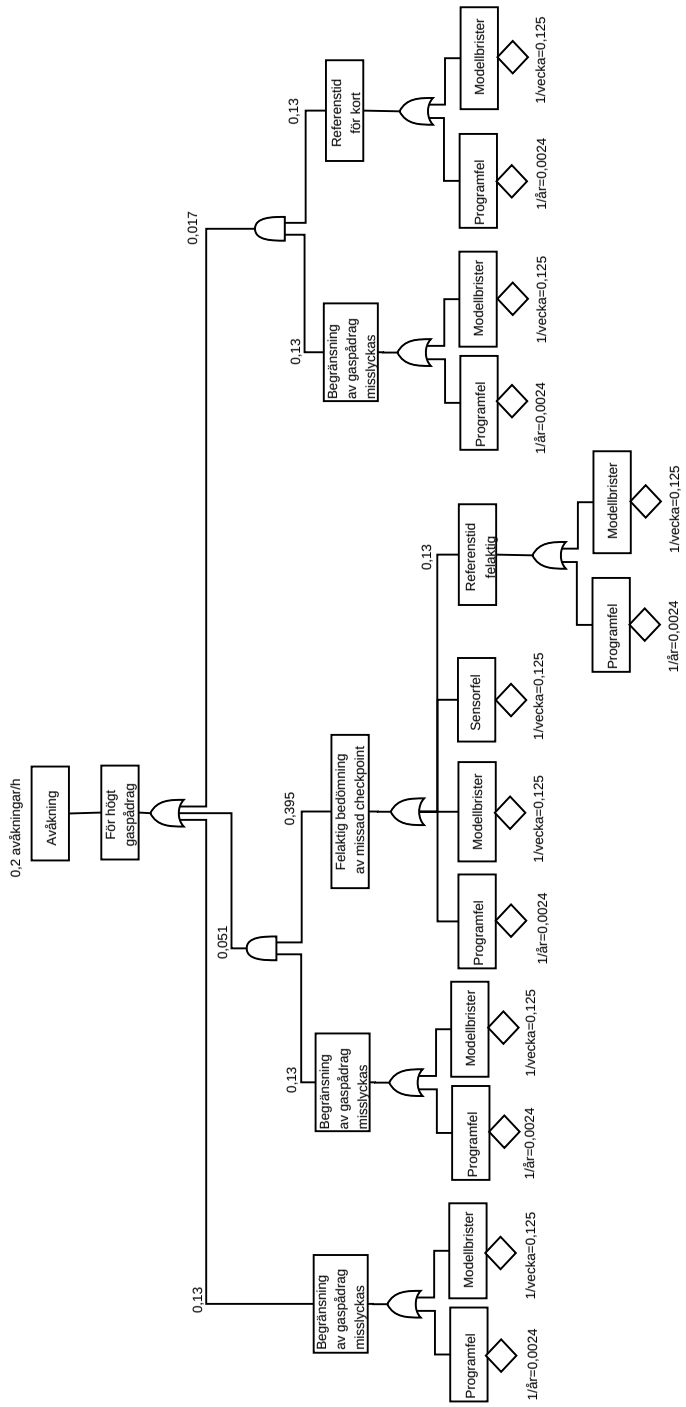
Den presenterade arbetsgången säger i avsnitt 4.3 att de övervakade delsystemen ska analyseras igen, denna gång med hänsyn taget till diagnossystemets positiva effekter, samt att ett nytt RPN ska räknas ut. Detta är redan gjort i projektet [5] och resultatet har presenterats i figur 4.4, så nästa steg blir här att utföra en FMEA på själva diagnossystemet. Ur FMEA-tabellen som illustreras i tabell 4.2, kan man via kolumnen *Felets effekt på systemet* utläsa att samtliga moduler har felbenägenheter som kan leda till avåkning - den oönskade topphändelsen för hela systemet. Detta då samtliga tre delsystem kan ha fel som alstrar för högt gaspådrag, vilket är den främsta anledningen till avåkningar. Kolumnen *frekvens* är graderad på en skala A-F, där A är högst frekvens och F lägst. Diagnossystemets händelser har graderats i de två kategorierna med lägst frekvens. Kolumnen *allvarlighetsgrad* är graderad från 1-4, där 1 är den allvarligaste konsekvensgraden. Denna används då risk för personskada föreligger. I detta fall anses diagnossystemet inte direkt kunna orsaka personskador, men dock direkta avkörningar. Därför har dessa fel klassats att vara av grad 2. RPN har beräknats som produkten av de två där A motsvaras av en etta och F av en sexa.

4.6.7 FTA över diagnossystemets negativa effekter

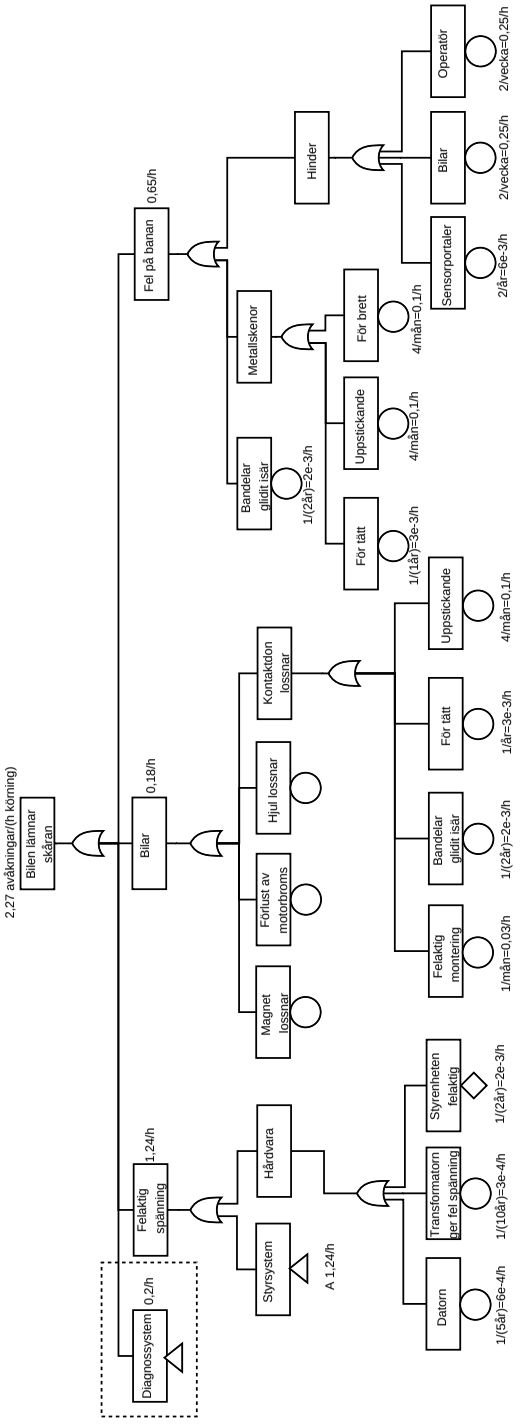
Felträdet för diagnossystemet byggs upp enligt de principer som redovisats i avsnitt 4.4. Först skapas en lista över de oönskade fel som diagnossystemet kan leda till och som samtidigt är intressanta för analysen (punkt 1). I detta fallet finns bara en händelse - bilen åker av banan. Varje objekt i denna lista är en topphändelse i ett felträd (punkt 2), i detta fallet alltså en enda - att bilen lämnar banan. Under topphändelsen byggs nu trädstrukturen upp steg för steg (punkt 3). Återigen poängterats att en mycket god kunskap om systemet är nödvändig för att kunna bygga upp korrekta felträd. Kunskap som kan hämtas ur FMEA är ett gott stöd, men är inte tillräckligt. Som det påpekades i det inledande avsnittet 4.6, är detta ett gedankenexperiment. Därför saknas nödvändig kunskap om programkoden som krävs för beräkning av felsannolikheter. Istället är de siffror som presenteras baserade på de resonemang som fördes i projektet [5], samt resonemang med handledare. Resonemanget är då att brister i respektive delsystem direkt orsakar felaktigt beteende. Enligt FMEA-tabellen 4.2, ligger felmoderna lågt i frekvens men högt i allvarlighetsgrad. RPN för felmoderna har medelhögt till högt värde (8-12). Med detta som bas, har felintensitet för modellbrister uppskattats till en gång per vecka, för programfel en gång per år, och för sensorfel en gång per vecka. Dessa tal kan anses vara lågt satta för att undvika att detta exempel ger för stora utslag.

Figur 4.5 visar felträdet för diagnosdelarna.

Med diagnossystemet tillagt i huvudträdet är resultatet figur 4.6. Händelsen att diagnossystemet alstrar fel är inlagt i trädet som *diagnossystem* längst till vänster direkt under toppnivån. En slutgiltig beräkning för bilbanans beteende med diagnossystem ger alltså en avåkningsfrekvens på 2,27 istället för 2,07. En skillnad på 10% är en tillräckligt stor skillnad för att belysa tanken att diagnossystemet i sig tillför negativa egenskaper till systemsäkerheten.



Figur 4.5: Felträd över bilbanans diagnossystem. Programfel och modellbrister adderar felbenägenhet till det totala systemet.



Figur 4.6: Felträd över bilbanan med diagnossystemets felbenägenhet adderad som den inramade händelsen *diagnossystem* längst till vänster i trädet. Diagnossystemet i sig har då tillfört osäkerhet och det totala antalet händelser per körd timme blir 2,27 istället för 2,07.

Kapitel 5

Diskussion

5.1 Slutsatser

Med hjälp av de två centrala metoderna i kvalitetsarbete, FMEA och FTA, är det möjligt att studera effekterna som uppkommer när ett diagnossystem införs. Arbetet sker med vedertagna metoder på ett strukturerat sätt. En mycket viktig slutsats är dock att dessa metoder kräver en mycket god systemkännedom för att vara effektiva.

Via beräkningar på ett befintligt system som ingår i ett större projekt, kunde teorierna verifieras. Diagnossystemet kan i sig själv tillföra en osäkerhet till systemet som övervakas. I exempelfallet var skillnaden 10%, en siffra som förvisso inte är statistiskt belagd, men som är tillräckligt stor för att bedömas som signifikant.

Det är svårt att göra en bra kvantitativ analys, alltså en analys med goda numeriska storheter för felsannolikheter. Processen att ta fram dessa felsannolikheter kräver även den god systemkännedom och förståelse för speciella analysverktyg. Särskilt problematisk är bedömningen av hur en operatör agerar som följd av vissa stimuli. Ett par metoder för att bedöma den mänskliga felfaktorn introduceras för ändamålet. Analys av programkod är även det en svår nöt att knäcka. För detta ändamål rekommenderas därför en kombination av ett flertal intressanta metoder.

Dock är den viktigaste slutsatsen att kvantitativ analys används i industrin och rapporten visar att kvantitativ analys kan visa de oönskade effekterna ett diagnossystem kan ge upphov till.

Litteraturförteckning

[1] R. Barlow, N.D. Singpurwalla, Z.W. Birnbaum och J.B. Fussel. *Reliability and fault tree analysis : theoretical and applied aspects of system reliability and safety assessment : papers presented at the Conference on ... held at the University of California, Berkeley, September 3-7, 1974 n.* Addison-Wesley, 1975.

[2] B. Bergman och B. Klefsjö. *Kvalitet, från behov till användning.* Studentlitteratur, andra upplagan edition, 1995.

[3] J-L. Lions et. al. Ariane 501 inquiry board report. internet, juli 1996. http://ravel.esrin.esa.it/docs/esa-x-1819eng.pdf.

[4] J.D. Musa, A. Iannino och K. Okumoto. *Software reliability, measurement, prediction, application.* McGraw-Hill, 1987.

[5] L. Nielsen, J. Biteus, E. Frisk, M. Krysander och G. Cedersund. Improving airplane safety by incorporating diagnosis into existing safety practice. Technical report, Institutionen för systemteknik, Linköpings universitet, 581 83 Linköping, Sverige, 9 Oktober 2003. preliminärt, att publiceras.

[6] M. Nyberg och E. Frisk. *Diagnosis and supervision of automated processes.* Linköpings tekniska högskola, Linköping, Sweden, 2001.

[7] O. Wiktorin och R. Ekholm. *H SystSäk E Försvarets handbok för systemsäkerhet.* Försvarsmakten, m7740-784851 edition, 1996.

[8] N. Ohlsson. *Towards effective fault prevention , An empircal study in software engineering.* Doktorsavhandling 522, Institutionen för datavetenskap, Linköpings Universitet, Linköping, Sverige, Maj 1998.

[9] A. Villemeur. *Reliability, availability, maintainability and safety assessment*, volume 1. John Wiley & sons, 1992.

[10] A. Villemeur. *Reliability, availability, maintainability and safety assessment*, volume 2. John Wiley & sons, 1992.

[11] E. Wu. Reliability of fault tolerant control systems, part 1, part 2. Technical Report WeA06-3,WeA06-4, 2003.

Copyright

Svenska

English