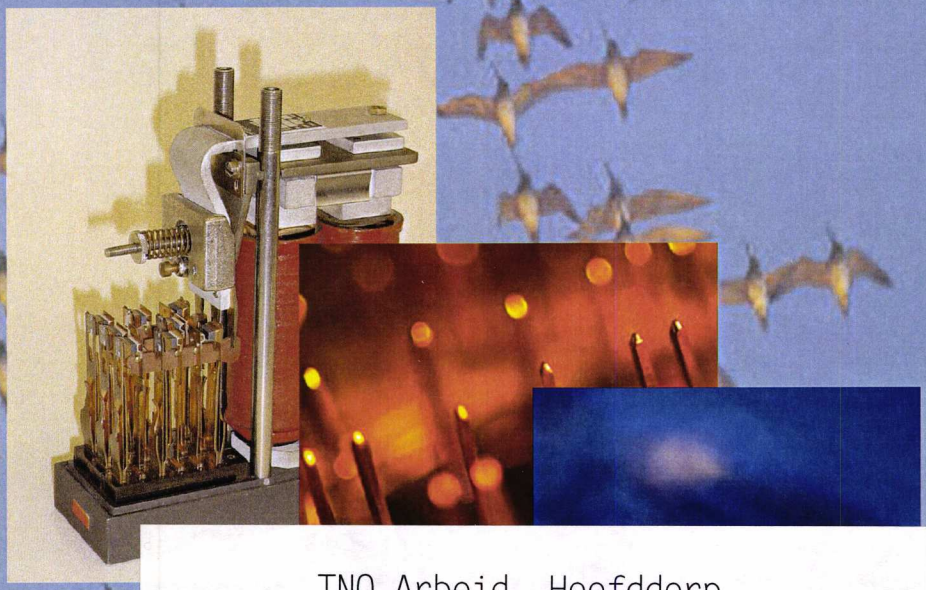
Safety Instrumented Systems in vogelvlucht



Auteurs:
Elly van den Bliek
Theo Logtenberg

afdeling Industriële Veiligheid
TNO Apeldoorn

Voorwoord

Ontwikkeling in de techniek is een voortgaand proces met vernieuwingen en soms blijken deze ontwikkelingen grote veranderingen met zich mee te brengen. De introductie van "Safety Instrumented Systems" kan als een grote verandering worden gezien. Het heeft het denken over beveiliging van processen op een heel ander spoor gezet.

SZW heeft dat erkend en heeft vervolgens TNO gevraagd een boekje samen te stellen waarin de belangrijke aspecten van "Safety Instrumented Systems" (afgekort SIS) worden toegelicht. Tevens is gevraagd een checklist samen te stellen waarmee de functionaliteit van een SIS globaal kan worden getoetst.

Dit boekje geeft de verzamelde kennis van een studieproject waarbij zowel de hardware als de software aspecten zijn bestudeerd. Het boekje geeft een tijdsbeeld. Veranderingen zijn zeker nog te verwachten. Industrie en onderzoekswereld houden zich nog volop bezig met zowel ontwikkelingen als analysemogelijkheden van deze complexe materie.

Bij de opzet en invulling van dit boekje is een belangrijke bijdrage geleverd door de Technische Universiteit van Eindhoven. Fotomateriaal is welwillend ter beschikking gesteld door Honeywell Safety Management Systems ('s-Hertogenbosch), Yokogawa Europe Industrial Safety Systems (Apeldoorn) en Mokveld Valves (Gouda).

Inhoudsopgave

1. *Inleiding*

Wat behandelen we in dit boekje?

Aandachtspunten

In dit boekje worden in vogelvlucht de achtergronden van *safety instrumented systems* (SIS) gegeven, door in te gaan op:

- De historie van de procesbeveiliging.
- Wat is de positie en rol van een SIS in de procesbeveiliging?
- Wat zijn de principes en uitvoeringsvormen?
- Welke eisen worden gesteld aan dergelijke systemen?
- Welke fouten kunnen er optreden?
- Hoe moet het ontwerpproces gebeuren volgens de standaard?
- Hoe kan het ontwerpproces worden beoordeeld?

Een en ander mondt uit in een tweetal verkorte checklists ter beoordeling van het ontwerpproces van *safety instrumented systems*.

Hoofdstukopbouw

Safety instrumented systems vormen geen gemakkelijk onderwerp. Daarom heeft dit boekje een zekere opbouw. De eerste hoofdstukken geven informatie over procesbeveiliging algemeen, een uitleg over SIS structuur, werkingsprincipes en opbouw, en de hardware en de software in een SIS.

Over vele van de aangeroerde zaken zijn reeds dikke boeken geschreven, waarmee een flinke boekenkast is te vullen. Het is aan de lezer zich hier verder in te verdiepen. Dit boekje is te zien als een korte introductie in het onderwerp, waarbij de gewenste omvang van het boekje beperkingen heeft opgelegd.

Onder de beoogde lezers voor dit boekje vallen alle mensen die in korte tijd een en ander te weten willen komen *over safety instrumented systems* om in het expertiseveld enigszins mee te kunnen praten en bij te kunnen blijven.

Nadere informatie

Voor meer details en (uitgebreidere) achtergrondinformatie is een lijst met aan te raden literatuur toegevoegd. Hierbij is alleen die literatuur aangegeven die als "standaard werk" of goede introductie bekend staan. Referenties zijn via voetnoten in de tekst met twee of drie letters aangegeven en het jaartal, bijvoorbeeld [AAR89] is een publicatie van Aarø, Bodsberg en Hokstad uit 1989.

Terminologie

In de wereld waar over *safety instrumented systems* wordt gesproken worden veel Engelse termen gebruikt. Deze worden in dit boekje naast de Nederlandse term cursief weergegeven. Sommige Engelse termen zijn niet vertaald in het Nederlands omdat daarmee de originele betekenis van de Engelse term verloren zou gaan. In het kader hieronder worden voor een algemeen begrip de gangbare Engelse en Nederlandse benamingen voor *safety instrumented systems* weergegeven.

Benamingen voor Safety Instrumented Systems

Engels
- *Safety systems*
- *E/E/PE safety-related systems*
  (E/E/PE = Electrical/Electronic/Programmable Electronic)
- *Emergency shutdown systems*
- *Safety interlock systems*
- *SGS = Safe Guarding System*

Nederlands
- *(instrumentele) veiligheidssystemen*
- *(instrumentele) beveiligingssystemen*
- *(instrumentele) beveiligingen*

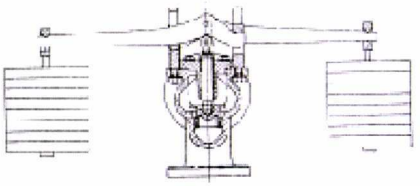
In hoofdstuk 11 is een overzicht gegeven van belangrijke afkortingen zoals die zijn opgenomen in de hoofdtekst en bijlagen van dit boekje.

2. *Procesbeveiliging*

2.1 Historie

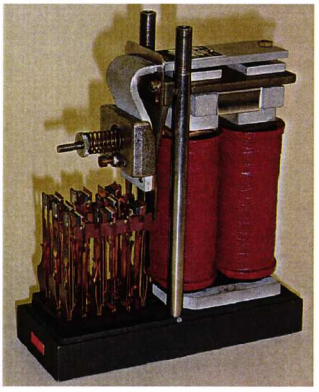
Beveiligen van systemen

Het beveiligen van processen was al een noodzaak bij de opkomst van de eerste stoommachines. De overdruk moest afgelaten kunnen worden ter voorkoming van het uit elkaar springen van de stoomketel. Hiervoor werd de mechanische beveiliging bedacht (zie figuur 2.1). Dit type beveiliging op de markt gebracht in1880 wordt nog steeds toegepast. (Eén bron vermeldt dat deze beveiliging zelfs al in 1812 werd toegepast).



Figuur 2.1 Voorbeeld mechanische overdrukbeveiliging.

Een andersoortige beveiliging die rond 1900 werd ontwikkeld is het relais en later het veiligheidsrelais (zie figuur 2.2). Een relais maakt of verbreekt een stroomkring indien het daarvoor een signaal krijgt. Relais zijn en worden nog steeds uitgebreid toegepast bij spoorwegovergangen maar ook bij industriële processen



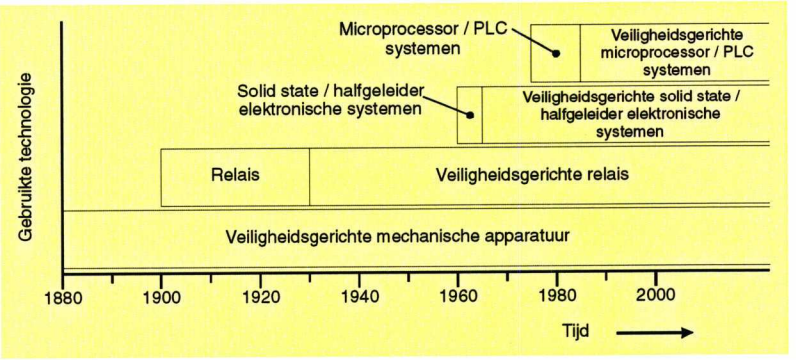
Figuur 2.2 Voorbeeld relaisbeveiliging.

De ontdekking van de halfgeleiders (transistors) was een volgende stap naar de "solid-state" beveiliging (een vaste niet aanpasbare regelkring). Men schrijft dan al het jaar 1960.

In de jaren zeventig heeft zich de ontwikkeling doorgezet naar microprocessor/ PLC-systemen. (PLC = Programmable Logic Controller). Een PLC kan een aantal te programmeren handelingen uitvoeren.

De PLC heeft ten opzichte van de eerder genoemde beveiligingen voordelen met betrekking tot flexibiliteit, beschikbaarheid, uitbreiding in functionaliteit, maar ook nadelen zoals een grotere complexiteit. Het wil overigens niet zeggen dat de PLC de mechanische beveiliging zonder meer kan vervangen, het is meer een aanvulling.

De verschillende ontwikkelingsstappen in een periode zijn weergeven in figuur 2.3. In de figuur is te zien dat de mechanische apparatuur nog niet is afgeschreven als beveiligingsmiddel en dat ook relais nog verkozen worden. Verder wordt duidelijk dat de op halfgeleider gebaseerde systemen en PLC's nog maar relatief kort worden toegepast.



Figuur 2.3 Historisch verloop van beveiligingen[1].

2.2 Introductie van software in processturing en beveiliging

SIS (Safety Instrumented System)

De mogelijkheid van het automatiseren van processen is min of meer gelijk opgegaan met de ontwikkeling van computersystemen. De PLC heeft daarbij een belangrijke plaats ingenomen. Eén of vaak meerdere PLC's hebben als hoofdtaak de sturing van het proces. Apart hiervan zijn de PLC's voor de beveiliging van het proces. Deze functioneren ***onafhankelijk*** van de processturing en grijpen in als een kritische waarde wordt overschreden. Deze beveiligings-PLC's worden aangeduid met de algemene term SIS (Safety Instrumented System) of ook wel met SGS (Safe Guarding System).

Omwenteling

De toepassing van een PLC betekent ook de introductie van software bij beveiliging. Het was een overgang van "hard wired" systemen (het zeer betrouwbare relais) naar een systeem waar nog niet echt ervaring mee was. Het is zeker als een omwenteling te zien. De ontwerpers hadden de nodige aarzeling om PLC-systemen toe te passen. De logische eis was dat de PLC-beveiliging net zo betrouwbaar moet zijn of mogelijk nog betrouwbaarder dan een relais.

1. Ontleend aan [ROU01].

Software in het dagelijks leven

Niet alleen in de procesindustrie is de overgang naar programmeerbare systemen te zien, ook in het dagelijks leven zijn talloze voorbeelden te vinden. In de eerste televisie bijvoorbeeld was geen sprake van software, maar in 1979 werd al een model uitgebracht dat 1 kilobyte aan software bevatte. In 1990 was dit reeds uitgebreid tot 64 kilobytes en in de hedendaagse televisie zit al gauw enkele megabytes aan software.

Ongevallen door software

Dat er redenen zijn om toch enig wantrouwen te hebben ten aanzien van geprogrammeerde systemen volgt uit een aantal ernstige ongevallen waarbij fouten in de software de oorzaak waren. De Volkskrant heeft daar al eerder een uitgebreid artikel aan gewijd. In bijlage 1 bij dit boekje is het artikel integraal overgenomen. Het artikel is al enkele jaren oud, het heeft echter aan relevantie nog niets ingeboet. Hieronder twee voorbeelden.

De Ariane raket

Voor de nieuwe Ariane 5 raket was dezelfde besturingssoftware gebruikt als voor zijn voorganger de Ariane 4, echter het ontwerp was wel wat aangepast. Dit was niet meegenomen in de software. Het gevolg was dat de raket na lancering van koers afging en vernietigd moest worden. Men spreekt hierbij wel over de duurste vuurpijl ooit (voor meer details zie bijlage 1: Lijst van software gerelateerde ongevallen).

Naverbrander

Bij het onderzoek naar het ontstaan van een explosie bleken drie zaken een rol te hebben gespeeld: een lek in het persluchtsysteem, een verkeerde afstelling van een drukmeter en een softwarefout. De software had de beveiliging moeten aansturen. Dezelfde fout bleek overigens in tal van vergelijkbare installaties te zitten.

Safety Instrumented Systems (SIS) niet alleen software

Hiervoor is de nadruk gelegd op de PLC-systemen als onafhankelijke beveiliging met software. Dat is niet geheel juist. De PLC meet, controleert en stuurt aan. Hierbij hoort ook hardware zoals de meetinstrumenten en een instrument dat de eigenlijke veiligheidsfunctie uitvoert, bijvoorbeeld een afsluiter.

De ontwikkeling en toepassing van *safety instrumented systems* heeft geleid tot het opstellen van standaarden. Voor de Nederlandse industrie is de Europese standaard IEC 61508 (spreek uit zes – vijftien – nul acht) van belang. De standaard geeft richting aan de diverse aspecten voor het tot stand komen, in bedrijf stellen en onderhoud van een *safety instrumented system*, in zowel technische als organisatorische zin.

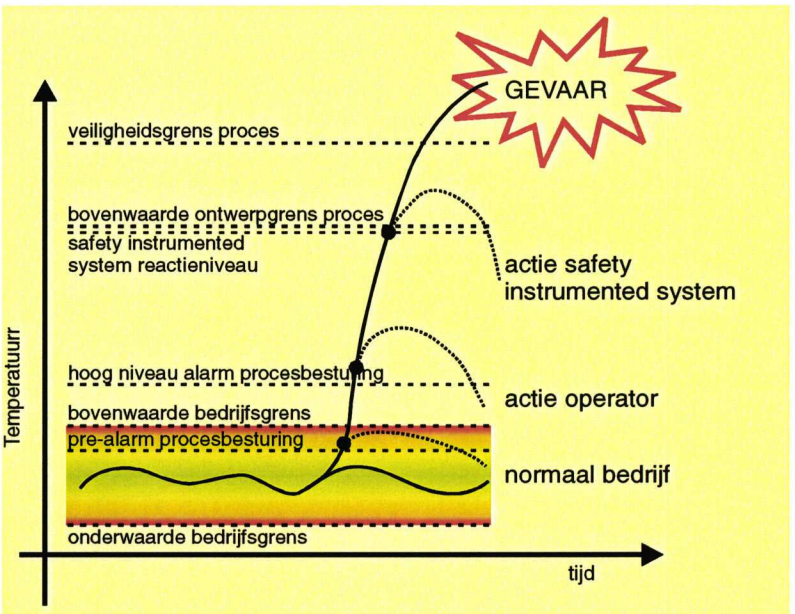
3. Plaats van SIS in de beveiligingslagen

In dit hoofdstuk wordt ingegaan op het begrip "beveiligingslagen van een proces", met name wanneer en hoe wordt ingegrepen bij een potentieel gevaar.

3.1 Procesbeheersing

Waarmee wordt een proces beheerst?

Stel we hebben een proces waarbij de beheersing van de temperatuur heel belangrijk is. Runaway-reacties (wegloopreacties, vaak polymerisatiereacties) zijn bijvoorbeeld zulke processen. In onderstaand figuur is een mogelijk verloop van zo'n reactie gegeven. Bij normaal bedrijf bevindt de temperatuur zich in het groen aangegeven gebied. Schommelingen kunnen zich voordoen, echter de procesregeling houdt het binnen de gestelde bedrijfsgrenzen.



Figuur 3.1 Mogelijk verloop van de temperatuur bij een wegloopreactie en systeemreacties[1].

Indien de temperatuur tijdens een reactie plotseling snel gaat toenemen dan zal een pre-alarm of waarschuwingsalarm worden gegeven. Een dergelijke snelle temperatuurtoename kan bijvoorbeeld gebeuren als de koeling uitvalt of als een operator te veel katalysator toevoegt.

De operator zal bij het pre-alarm actie moeten ondernemen. Het is echter mogelijk dat de temperatuur te snel stijgt of de operator te langzaam reageert.

1. Ontleend aan [ROU01].

Indien de operator niets doet zal een automatische ingreep moeten plaats vinden door het SIS (Safety Instrumented System).
Het SIS kan dan bijvoorbeeld een quenching (blussing) in werking zetten (toevoegen van een grote hoeveelheid koud verdunningsmiddel) of het reactiemengsel afvoeren naar een zogenaamd "knock-down" vat.

Is ook de ingreep van het SIS niet voldoende en stijgt de temperatuur nog verder en (daarmee ook de druk) dan moet de laatste beveiliging in actie komen. Dat is de mechanische of overdrukbeveiliging, die voorkomt het exploderen van de reactor door af te blazen. Hierdoor komen de reagerende stoffen echter wel naar buiten. Het hangt van de stoffen af wat dat voor gevolgen kan hebben, bijvoorbeeld als het zeer brandbare, giftige dan wel explosieve stoffen betreft.

Wat bepaalt nu de veiligheid van een proces?

In de eerste plaats is het proces zelf bepalend voor de veiligheid: welke stoffen worden toegepast, welke handelingen, temperaturen, drukken en hoeveelheden. Daarnaast is het van belang hoe het proces kan worden geregeld. Volledig automatisch of met veel ingrepen van een operator.

Voorbeeld ingrijpen van een operator in een proces

- *Een bedrijf heeft een proces waarin twee deelprocessen centraal staan, te weten een extractieproces en een kookproces. De omstandigheden van het extractieproces worden bepaald door het ruwe materiaal dat wordt aangevoerd (verontreinigingen en hoeveelheid extraheerbare stof). De aanvoer is niet constant. De operator moet voor dit procesonderdeel telkens aanpassingen doen. Het geeft in het algemeen een wat onrustig beeld van het procesverloop met de mogelijkheid dat soms instellingen niet tijdig of verkeerd worden gedaan.*
- *Bij het kookproces liggen de zaken heel anders. Het product is constant van samenstelling en het kookproces heeft een theoretische grondslag die heel precies kan worden geprogrammeerd. Hier gaat alleen wat mis als de techniek het laat afweten.*

Verhoging van de veiligheid: Streven naar inherente veiligheid

Een eerste uitgangspunt bij een te beveiligen proces is het streven naar inherente veiligheid. Men onderzoekt of voor een proces minder gevaarlijke alternatieven voorhanden zijn. Hierbij kan bijvoorbeeld aandacht worden besteed aan procestemperatuur (lagere procestemperatuur mogelijk gemaakt door het gebruik van een katalysator), procesdruk en kleinere bewerkings- en opslaghoeveelheden. Het eindresultaat van het onderzoek moet zijn dat wordt vastgesteld onder welke omstandigheden de energie-inhoud van het proces zo klein wordt dat het niet buiten zijn omhulling kan treden. Het zal echter in veel gevallen niet mogelijk zijn het proces zodanig veilig uit te voeren dat geen extra veiligheid nodig is.

3.2 Beveiligingslagen

In het gegeven voorbeeld was sprake van handeling van de operator, ingrijpen van het Safety Instrument System en het afblazen via de mechanische beveiliging. Men geeft dit wel aan met beveiligingslagen. Deze drie lagen tezamen bepalen **de kans** dat er iets mis gaat. Hierin heeft zowel de operator als het Safety Instrumented

System alsmede de mechanische beveiliging een bijdrage in uiteindelijke veiligheidsniveau.

Begrip kans

Kans is de uitdrukking voor het aandeel in het totaal aantal gebeurtenissen dat er iets specifieks gebeurt. Vergelijk een dobbelsteen: de kans dat men een zes gooit is één-zesde. Naast kans wordt ook frequentie gebruikt, dat geeft aan hoe vaak iets in een zekere tijdsperiode gebeurt.

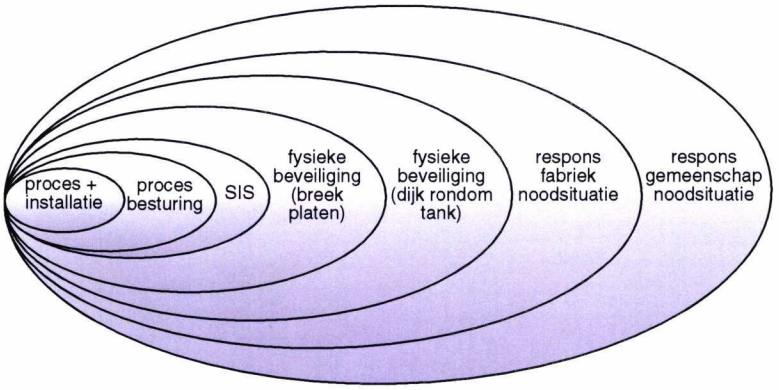
Meer lagen

Behalve de genoemde drie lagen die de kans op een gevaar verkleinen worden nog drie lagen onderkend. Dat zijn de effectverkleinende lagen. Bijvoorbeeld een dijk om een opslagtank beperkt bij een groot lek van de tank het verdampingsoppervlak en gaat verspreiding over het terrein tegen.

Het zal duidelijk zijn dat de inzet van de eigen bedrijfsbrandweer of de regionale brandweer een verlagend effect hebben, dat wil zeggen: men gaat door tijdig ingrijpen tegen dat een brand zich uitbreidt.

Het geheel van de kansverlagende en effectverlagende maatregelen worden aangeduid met het *Layer of Protection* principe[1]. De lagen worden als onafhankelijk van elkaar gezien.

Een schematische weergave van de beveiligingslagen staat in figuur 3.2 Alle lagen **tezamen** zorgen voor de procesbeveiliging. De lagen bestaan uit de te onderscheiden technische en organisatorische maatregelen zoals hiervoor genoemd.



Figuur 3.2 Voorbeeld van de lagen van veiligheid rondom een proces.

1. Literatuurreferentie voor Layer of Protection principe [CCP01].

Wat doet een SIS?

Een SIS meet onafhankelijk van de processturing een aantal procesvariabelen, controleert deze (vergelijkt een gemeten waarde met ingestelde veiligheidswaarde), geeft zonodig een alarm en grijpt in als dat nodig is. De procesvariabelen kunnen bijvoorbeeld zijn de temperatuur, de druk, het niveau in een tank of stroming in een leiding.

Voorbeeld toepassing SIS

Bij een overslagproces is het van belang dat het niveau in de tank niet te hoog wordt. Een voorwaarde kan ook zijn dat de temperatuur beneden een zekere waarde blijft. Deze twee variabelen kunnen door een SIS worden bewaakt. Daarnaast kan de stroomsnelheid worden bewaakt. Een te hoge stroomsnelheid kan een indicatie van een lek zijn. Tevens kan worden gecontroleerd of afsluiters in de goede stand staan. Veel invoerparameters die de operator uiteindelijk helpen het overslagproces veilig uit te voeren.

Een andere functie van een SIS

In het voorgaande had het SIS een functie in de beheersing van het proces. Bij het overschrijden van een kritische waarde werd automatisch ingegrepen. Het SIS verlaagt dan de kans op een ongewenste gebeurtenis. Een SIS kan ook een rol spelen in het verlagen van het effect, bijvoorbeeld als het SIS onderdeel uit maakt van een op zich zelf staand beveiligingssysteem, bijvoorbeeld voor het blussen van een brand. Stel een ruimte is beveiligd met infrarood sensors voor het waarnemen van een brand. Het signaal gaat naar een PLC, die vervolgens een blussing in werking stelt.

Begrippen 'de-energise to trip' en 'energise to trip'

Het stoppen van een proces en het op gang brengen van een proces door een SIS brengt ons naar twee belangrijke begrippen: *de-energise to trip* en *energise to trip*.

De-energise to trip betekent dat de energietoevoer wordt gestopt. *Energise to trip*, betekent dat juist energie toegevoerd voor een veilige actie. De energie kan in de vorm van elektriciteit of perslucht zijn.

Voorbeeld: De-energise to trip

- *Een afsluiter wordt opengehouden door persluchtdruk. Indien de druk wegvalt (wegvallen energie) sluit de afsluiter automatisch door veerdruk.*
- *Een ander voorbeeld is bij de beveiliging van spoorwegovergangen. Indien bij een spoorwegovergang de energievoorziening wegvalt, gaan de overwegbomen half sluiten. Ze zakken door eigen gewicht naar beneden en worden niet meer omhoog gehouden. Niet sluiten zou een veel gevaarlijker situatie opleveren.*

Voorbeeld: Energise to trip

- *Een noodafsluiter is voorzien van een motor die elektrische voeding nodig heeft voor het uitvoeren van een sluitactie. Hier wordt dus energie toegevoerd.*

Het '*de-energise to trip*' in een proces heeft meestal de voorkeur, aangezien in praktijk het tijdens een noodsituatie makkelijker en betrouwbaarder is de energietoevoer af te sluiten dan energie toe te voeren. Bovendien resulteren veel defecten in een SIS juist in het afsluiten van de energietoevoer, waardoor het proces juist in een veilige situatie wordt gebracht.

Overige taken van een SIS

Het SIS heeft vaak niet alleen een veiligheids-functie. Het systeem kan ook gemeten waarden vastleggen alsook de acties van de operator. Dit wordt omschreven met *sequence-of-events recording* (loggen en bewaren van gebeurtenissen).

Begrenzing

Een SIS moet onafhankelijk zijn van zijn omgeving, evenals van andere veiligheidsmaatregelen (zoals fysieke beveiligingen en procedures). Dit betekent ook dat alle inkomende signalen aan de volgende criteria moeten voldoen:

- Correct, in de zin van: juiste indicatie van de procesconditie
- Valide, in de zin van: binnen het bereik van de sensoren

De uitgangssignalen moeten eveneens correct zijn, dat wil zeggen de signalen moeten de ingrijpende procesonderdelen juist aansturen (bijvoorbeeld een snelaansluiter). Het is daarbij van belang dat geen signaal wordt gegeven die een onterechte (sluit)actie tot gevolg heeft (een zogenaamde "false-trip"). Het zal duidelijk zijn dat dit eisen stelt aan de uitvoeringsvorm van de diverse componenten. Invloeden van buiten, bijvoorbeeld magnetische pulsen of indringing van vocht waardoor kortsluiting zou kunnen ontstaan, moeten uitgesloten worden.

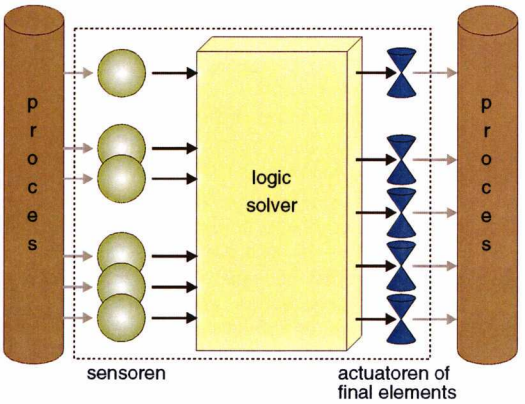
4. *Opbouw van safety instrumented systems*

De plaats van een *safety instrumented system*[1] in de beveiliging is in hoofdstuk 3 reeds globaal aan de orde geweest. Hieronder wordt de algemene opbouw van een SIS besproken, de verschillende uitvoeringsvormen en samenstellende elementen.

4.1 Algemene opbouw van een safety instrumented system

Opbouw SIS

Een SIS verricht een aantal metingen waarvoor sensoren nodig zijn. De gemeten waarden moeten worden beoordeeld en dat gebeurt in een zogenaamde *logic solver*. Afhankelijk van de gemeten waarde wordt een actie gedaan door actuatoren, meestal ook aangeduid als *final elements*. In onderstaande figuur is de vereenvoudigde opbouw van een SIS geschetst.

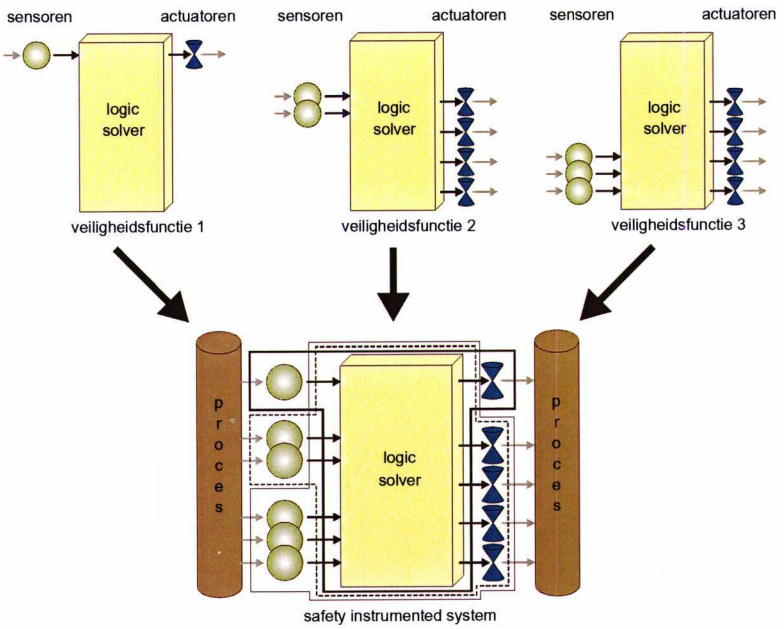


Figuur 4.1 Algemene opbouw van een SIS.

Meerdere veiligheidsfuncties

Zoals uit de voorgaande figuur 4.1 blijkt zijn er meerdere sensoren en actuatoren die verschillende veiligheidsfuncties kunnen hebben. Dit is verder uitgebeeld in figuur 4.2. Een samenstel van een sensor (bijvoorbeeld een temperatuuropnemer) en een actuator heeft veiligheidsfunctie 1. Een samenstel van twee andere sensoren (bijvoorbeeld twee temperatuuropnemers of een temperatuuropnemer en een drukopnemer) en vier actuatoren zijn ondergebracht in veiligheidsfunctie 2. Drie andere sensoren en de reeds eerder aangegeven vier actuatoren verzorgen veiligheidsfunctie 3. Het geheel van de verschillende veiligheidsfuncties wordt in één SIS samengebracht.

1. Zie ook de literatuurreferenties [GOB92, GRU98, ROU01].



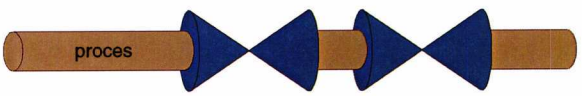
Figuur 4.2 Veiligheidsfuncties gecombineerd in één SIS.

Meervoudige meting

Sensoren kunnen op verschillende plaatsen in het proces zijn aangebracht, maar het is ook mogelijk dat bijvoorbeeld drie sensoren op één plaats één procesvariabele volgen. Men krijgt dan drie signalen. Deze moeten normaal gesproken nagenoeg identiek zijn. Indien één van de waarden afwijkt door een technische reden dan zijn er nog twee andere meetwaarden waar men gebruik van kan maken. Als één van de sensoren een afwijking aangeeft behoeft men het proces niet stil te leggen, echter als er tenminste twee van de drie aangeven dat een kritische waarde wordt overschreden dan zal het SIS ingrijpen. De meervoudig meting wordt toegepast als men zeker wil weten dat de gemeten waarde correct is. Het uitschakelen van een proces gebeurt dan niet op een toevallige waarde.

Actuator meervoudig uitgevoerd

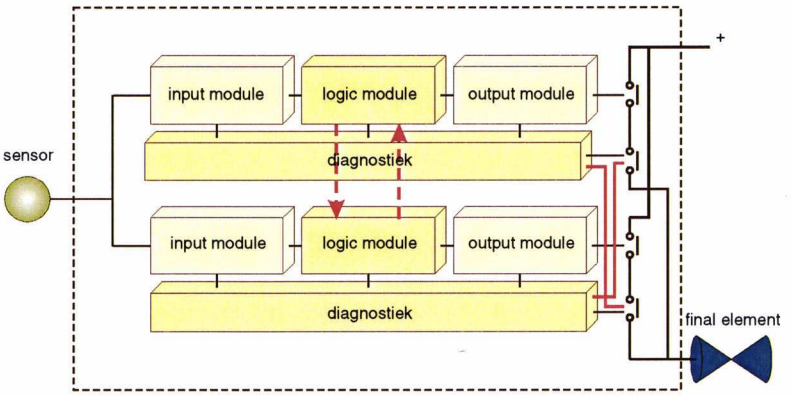
Behalve de sensoren kunnen de actuatoren ook meervoudig zijn uitgevoerd, zie onderstaand figuur. Indien een afsluiter niet volledig dicht gaat dan is er nog een tweede afsluiter die de gewenst sluitactie dan nog wel uitvoert. Eén van de twee moet het dan doen en dat geeft meer zekerheid.



Figuur 4.3 Meervoudige uitvoering van een actuator.

Meervoudige uitvoering logic solver

In de voorgaande figuren is telkens de logic solver als één eenheid afgebeeld, maar het bestaat uit verschillende componenten. Een logic solver heeft in ieder geval een input module voor het invoeren van gegevens en een output module voor het uitvoeren van signalen. Daar tussen in zitten de logic module (de processor). De gehele set van kan weer een controlerende laag hebben, aangeduid met diagnostiek. Deze opbouw kan meervoudig zijn uitgevoerd, waarbij over en weer wordt gecheckt. Hiervoor kunnen fraaie softwarematige algoritmen worden uitgewerkt.



Figuur 4.4 Meerdere componenten in een logic solver.

Het meervoudig uitvoeren van componenten in een logic solver wordt niet zo maar gedaan. Het wordt vooral toegepast voor PLC's die een hoge betrouwbaarheid moeten hebben, bijvoorbeeld beveiligings-PLC's.

In het algemeen wordt de meervoudige uitvoering van componenten in een SIS aangeduid met redundantie. Het begrip redundantie is op zich eenvoudig maar in de uitvoeringsvormen kunnen deze nogal complex zijn. Een meer gedetailleerde uitwerking staat in bijlage 2. Daarin wordt ook ingegaan op het begrip *voting* (beslissen op basis van meer binnenkomende signalen).

4.2 Samenstellende elementen nader bekeken

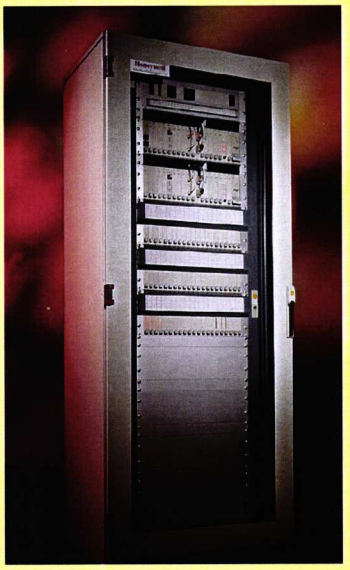
Typen sensoren

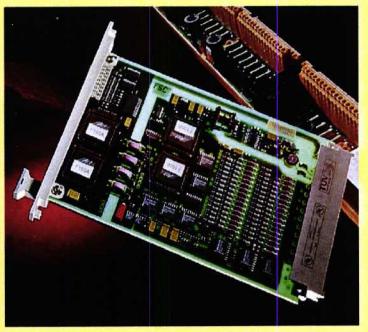
Hiervoor is al regelmatig de sensor als onderdeel van een SIS genoemd. De sensoren kunnen worden onderscheiden in analoge en digitale sensoren. Analoge geven een zekere waarde van een procesparameter en digitale sensoren geven een soort ja/nee signaal. De analoge sensoren worden "transmitters" genoemd en de digitale sensoren worden aangeduid met "switches".

Indien voor een zekere procesparameter (temperatuur of druk) meerdere sensoren worden gebruikt kunnen deze "diverse" worden uitgevoerd dat wil zeggen dat de sensoren fysiek niet exact hetzelfde zijn uitgevoerd, bijvoorbeeld zowel analoog als digitaal uitgevoerd.

Logic Solver

De logic solver is het eigenlijke brein van een SIS. De logic solver neemt de beslissingen gebaseerd op de signalen van de sensoren. In de onderstaande figuur zijn twee plaatjes opgenomen voor een indruk van een opbouw van een logic solver.





Links de kast met de elektrische aansluitingen en elektronica, zoals de in- en uitvoerkaarten, omzetters etc. De foto boven toont de eigenlijke logic solver.

Actuatoren

Actuatoren zijn de installatieonderdelen die de eigenlijke veiligheidsactie uitvoeren. Ze sluiten een leiding af, starten generatoren of stoppen pompen.

In de figuur hieronder is een voorbeeld van een actuator te zien. De actuator is een afsluiter en bestaat uit de eigenlijke afsluiter en een stuurmechanisme voor bediening.

4.3 Software in safety instrumented systems

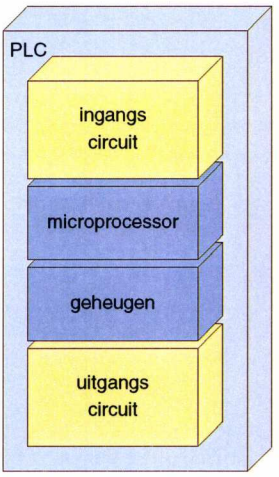
De logic solver zoals besproken is meestal een *PLC* een: *Programmable Logic Solver*. De aanduiding *Programmable* zegt het al, het instrument is programmeerbaar en bevat dus software. Overigens begint men bij sensoren ook reeds software te gebruiken.

Hieronder wordt op de structuur en de werking van een PLC ingegaan en worden twee verschillende softwaremodules besproken.

4.3.1 De PLC

Structuur van PLC's

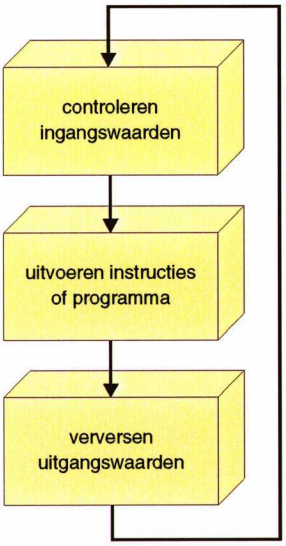
Een PLC bestaat globaal uit een ingangscircuit, een microprocessor, een geheugen en een uitgangscircuit. Het ingangs- en uitgangscircuit kunnen weer bestaan uit meerdere ingangen en uitgangen. De globale structuur van een conventionele PLC wordt weergegeven in onderstaand figuur. (Opmerking: Het principe van de PLC is voor besturing en beveiliging hetzelfde. De besturing en de beveiligings-PLC zijn wel altijd gescheiden van elkaar.)



Figuur 4.5 Globale structuur van een conventionele PLC.

Scancyclus

De microprocessor en het geheugen zijn de plaatsen die met de software te maken hebben. Het proces gaat als volgt. Het programma van de processor zal eerst alle ingangssignalen stuk voor stuk uitlezen en controleren. Daarna wordt het programma doorlopen. Het programma bestaat uit een aantal instructies die de uitgangswaarden bepalen. De uitgangswaarden gaan naar het uitgangscircuit; men spreekt dan van verversing of genereren van de uitgangswaarden. Deze drie stappen vormen de zogenaamde **scancyclus** van een PLC. Wanneer deze drie stappen zijn doorlopen begint de cyclus opnieuw: lezen ingangen, uitvoeren programma, verversen uitgangswaarden. Het continue proces wordt weergegeven in figuur 4.5.



Figuur 4.6 Doorlopen van één PLC-cyclus.

Responstijd

De tijdsduur van één cyclus wordt ook wel responstijd genoemd. De uitvoering van een cyclus kost enige tijd (milliseconden). In deze tijd kunnen de ingangswaarden zijn veranderd, dat wordt door de PLC pas gezien als de ingangwaarden opnieuw worden gelezen. In bepaalde gevallen kan het belangrijk zijn dat een belangrijke verandering eerder tot een actie moet leiden. De fabrikant van een PLC kan het programma daarop inrichten.

Typen software

De software aanwezig in de logic solver van het *safety instrumented system* kan worden onderverdeeld in twee soorten *embedded* software en *application software*.

Embedded (ingebed, ingebakken of ingevroren) software is de 'vaste' software in de PLC en is bedoeld als besturing (en programmeeromgeving). Daarnaast kan een hoeveelheid software zijn ingebakken die een zekere functie uitvoert, maar omdat het embedded is kan deze software niet meer worden aangepast. Dit ligt anders voor "Application" (toegepaste) software waar voor de veiligheidsfunctie nog wel aanpassingen mogelijk zijn; bijvoorbeeld het aanpassen van set-points (de kritische waarde waarop wordt gecontroleerd, het invoeren van meer controlepunten of het aanpassen van specifieke relaties van parameters.

Voorbeeld embedded software

In de Noordzee bevinden zich een aantal meetinstrumenten die metingen verrichten aan getijden en weersomstandigheden. Bij een bepaald verloop zal alarm moeten worden gegeven voor het nemen van maatregelen zoals het afsluiten van stormvloedkeringen. De software in deze instrumenten is embedded en kan niet zomaar worden aangepast.

Application software kan dus worden veranderd maar hiervoor moeten strikte procedures worden afgesproken. Indien een afnemer het programma naar eigen inzichten aanpast kan de leverancier niet meer instaan voor de goede werking. De leverancier moet daarentegen als hij om aanpassingen wordt gevraagd er ook zeker van zijn dat de aanpassing niet de functionaliteit aantast. Om dit zeker te stellen moet de software uitvoerig worden getest.

Meer complexiteit in software

In het verleden werd de de programmering in een SIS gedaan via eenvoudige opeenvolgende stappen ook wel aangeduid als ladderlogica. We willen echter meer: meer controleren, meer relaties leggen tussen meetwaarden, het bijhouden van gebeurtenissen en informatie naar de operator. De functionaliteit kan ook worden getest via een te programmeren protocol. Al met al betekent het ook dat de software complexer wordt.

4.3.2 Software modules

Voor het maken van de programmering van de logic solver kan de programmeur gebruik maken van reeds bestaande commercieel verkrijgbare modules die hij aan elkaar koppelt of hij programmeert het van begin tot eind helemaal zelf.

Men onderscheidt twee soorten modules met de voor Nederlandse begrippen wat vreemde engelse acroniemen te weten: *COTS* en *SOUP*[1].

Module COTS

COTS staat voor *Commercial off-the-shelf software* en is software die reeds bestaat en waar de SIS ontwerper niets aan kan veranderen. COTS is vaak generiek, kan functies bevatten die niet nodig zijn, en kan, aangezien het commercieel verkrijgbare en veelgebruikte software betreft, onderhevig zijn aan veranderingen door de leverancier. Voorbeelden van COTS zijn: software die een onderdeel vormt van een programma zoals, berekeningsalgoritmen, en programmadelen voor aansturing (*drivers*). Ook kunnen dat zijn zelfstandige programma's (*compilers*), high-level programma's bijvoorbeeld besturingssystemen en complete geïntegreerde systemen zoals alarm systemen.

Module SOUP

SOUP staat voor *Software of unknown pedigree*. Hiermee bedoelen we dat we niet goed weten wat de historie van ontwikkeling is: voor welke toepassing oorspronkelijk ontwikkeld en welke aanpassingen zijn gedaan in de verschillende versies. Het geeft dan wat minder vertrouwen, zoals de naam eigenlijk ook suggereert, voor een goede werking van een toch zo belangrijk instrument als een SIS. Indien men niet anders heeft dan zal men hierop aangewezen zijn.

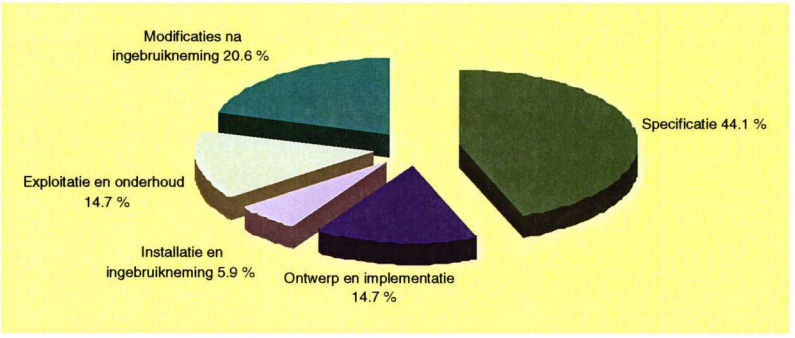
1. Beschrijving typen modules in [HSE01],

5. Fouten in een SIS

Fouten in een *safety instrumented system* kunnen tot gevolg hebben dat het SIS niet kan ingrijpen wanneer het proces daar om vraagt waardoor de benodigde risicoreductie niet wordt gerealiseerd. In dit hoofdstuk worden de factoren genoemd die kunnen leiden tot fouten in SIS. Daarna wordt ingegaan op faalvormen, typen fouten en een mogelijke verdeling van deze fouten en software fouten.

5.1 Invloedsfactoren op safety integrity

Factoren die invloed hebben op de *safety integrity of bedrijfszekerheid* van *safety instrumented systems* zijn de uitvoeringsvorm en de betrouwbaarheid van de hardware, evenals de betrouwbaarheid van de software. Hierin spelen het ontwerpproces het gebruik van het systeem en menselijke factoren[1] een rol. Analyses van eerder gebeurde ongelukken wijzen erop dat oorzaken liggen een combinatie of keten van deze factoren[2]. De verdeling van fouten van een controlesysteem zoals vastgesteld in een onderzoek van de HSE (Health and Safety Executive - UK) staat in onderstaande figuur 5.1.



Figuur 5.1 Oorzaken van fouten in control system[3].

Zoals uit de figuur blijkt liggen fouten vooral in specificatie en modificaties na ingebruikname. Een correcte specificatie is dus van groot belang. Het betekent ook dat men van tevoren goed moet onderkennen wat van het systeem wordt gevraagd. Hiervoor zullen de gebruikelijke risico- en veiligheidsanalyses inzicht moeten geven, bijvoorbeeld FMECA of andere technieken. Een beschrijving van deze technieken is gegeven in bijlage 3.

1. Zie voor de invloedsfactoren op de fouten ook de volgende literatuurreferenties [KLE94, HSE95, NEU95].
2. Verdere informatie over invloedsfactoren is te vinden in [PER84, HSE95].
3. Figuur is ontleend aan [HSE95].

5.2 Fouten in hardware

Fysieke faalvormen

In een SIS kunnen zich verschillende faalvormen voordoen bijvoorbeeld een kortsluiting in de bedrading, het lekken van een klep, het driften van een sensor, een invoerkaart die verkeerde signalen afgeeft.

Typen fouten

Het falen van een SIS in een aantal typen fouten in te delen; te weten random, common cause en en systematische fouten. De omschrijving van deze fouten is als volgt:

- *Random* of 'normale' fouten zijn fouten die op willekeurige tijdstippen optreden. Het kan op elk moment tijdens een gebruiksperiode plaats vinden. Dit in tegenstelling to slijtage of corrosie fouten, die na een zekere periode optreden. Men spreekt dan van veroudering.
- *Common cause* fouten zijn fouten die effect hebben op meerdere componenten en een gemeenschappelijke oorzaak hebben.
- Systematische fouten zijn fouten in het ontwerp, assemblage of vanwege complexiteit van het systeem . Deze fouten zijn alle fouten die niet kunnen worden gekarakteriseerd als random en common cause fouten.

Random fouten

Blikseminslag komt regelmatig voor. De mogelijkheid van een inslag in een SIS met belangrijke gevolgen is wat betreft kans te voorspellen, alleen niet wanneer het gebeurt. Het treedt random (willekeurig) op.

Common cause fouten

Stel we hebben in een leiding vervuild vervuild gas dat corrosie veroorzaakt in twee in serie geplaatste kleppen. De twee kleppen in serie waren ter verhoging van de redundantie, echter de corrosie tast de kleppen in dezelfde mate aan en beide zullen dan niet of onvoldoende sluiten indien dat nodig is. Dit is een common cause fout

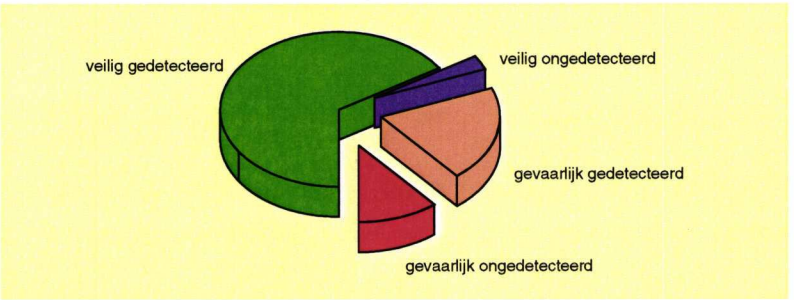
Systematische fouten

Een sensor moet op 2 psi worden ingesteld voor het afgeven van een alarmsignaal. De installateur of operator is in de war met de eenheden en stelt de sensor af op 2 bar. Het betekent dat ondanks dat het systeem in principe goed zou kunnen werken het toch zijn functie niet vervult. Het zou overigens wel zo goed zijn als de verkeerde instelling met een testmethode zou zijn vastgesteld.

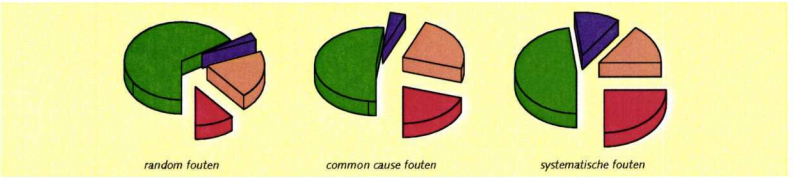
Detecteren van fouten

Een fout in een SIS kan geen direct gevolg hebben voor het functioneren van een SIS, (men spreekt dan van een veilig gevolg) of wel de functie belangrijk aantasten en dan spreekt men van een gevaarlijk gevolg. Belangrijk is of men de fout detecteert. Er kunnen ook fouten optreden die geen gevolgen hebben voor de beschikbaarheid van het SIS. Deze worden *no-effect failures* genoemd.

Een mogelijke verdeling van de fouten in veilig, gevaarlijk, gedetecteerd en ongedetecteerdfaalvormen is ter illustratie in figuur 5.2 weergegeven. Een verdere onderverdeling van de fouten in typen fouten is in figuur 5.3 weergegeven.



Figuur 5.2 Mogelijke verdeling van fouten.



Figuur 5.3 Nadere onderverdeling van de fouten van figuur 5.2 in typen fouten.

Uit de figuren is af te lezen dat het gezamenlijk aandeel veilig gedetecteerd en veilig ongedetecteerd het grootst is bij alle drie de typen fouten. Gevaarlijk ongedetecteerd heeft echter toch nog wel een behoorlijk aandeel, zeker bij de systematische fouten.

Diagnostics en testen

Om fouten te detecteren in het *safety instrumented system* worden (diagnostische) testen uitgevoerd. Deze testen kunnen grofweg worden verdeeld in twee soorten:

- On-line diagnostics (continue controle op het aanwezig zijn van fouten)
- Off-line of periodieke testen

Met name de modernere en de programmeerbare componenten hebben *on-line diagnostics*. Het checken vindt dan praktisch continu plaats (bijv. om de 50 milliseconden). *Diagnostics* zorgen er voor dat het falen van een component vrijwel onmiddellijk gedetecteerd wordt, waarop eventueel actie kan worden ondernomen (zoals vervanging of reparatie). Deze *diagnostics* kunnen echter niet alle fouten detecteren, doordat de *diagnostics* mogelijk zelf falen, of doordat de detectie niet op bepaalde foutmodes let of kan letten. Een voorbeeld van *diagnostics* is een zogenaamde *watchdog*. Een *watchdog* houdt bijvoorbeeld in de gaten of een bepaald signaal verandert zoals afgesproken in het ontwerp. Verandert dit signaal niet, dan zorgt hij voor een (detectie)alarm.

Aangezien niet alle fouten gedetecteerd worden of kunnen worden door de *diagnostics*, vinden daarnaast periodieke testen plaats. Hiervoor zal het proces geheel of gedeeltelijk moeten worden stilgelegd. Bij een gedeeltelijk stilleggen van een proces zullen delen van het SIS die niet worden getest moeten blijven functioneren en zal het te testen deel moeten worden overbrugd (ook wel aangeduid met override).

Ondanks de uitgebreide en gedetailleerde testprotocollen en testprocedures voor periodieke testen, zal nog steeds een klein percentage van de fouten niet gevonden worden of kunnen worden.

Reparatie fouten

Bij het constateren van fouten moet een reparatie plaats vinden. Wanneer en hoe de reparatie moet plaats vinden, hangt af of de ingreep groot of klein is en of daarvoor het proces stil moet worden gelegd. Een vervanging van een I/O kaart kan meestal tijdens bedrijf gebeuren. Voor een vervanging van een sensor zal het proces moeten worden gestopt. Dit geldt ook voor een groot probleem in de logic solver.

Het repareren van een SIS kan mogelijk weer nieuwe fouten introduceren, en derhalve blijft testen belangrijk.

5.3 Fouten in software

Voor het nagaan van de betrouwbaarheid van de software worden momenteel enkel testen of kwalitatieve analysemethoden worden gebruikt[1].

Onderverdeling methoden

De methoden die momenteel gebruikt worden om de betrouwbaarheid van software te beoordelen kunnen als volgt worden onderverdeeld[2]:

- Naar aanpak: testmethoden en kwalitatieve analyses
- Naar focus: beoordeling van de software of van het software ontwikkelproces

Beide methoden hebben tot doel:
a) Het voorkomen van fouten
b) Het verwijderen van fouten
c) Het detecteren van fouten
d) Het zorgen voor een hoge mate van fouttolerantie

Meestal wordt een combinatie van de vier hier beschreven aanpakken toegepast om een aanvaardbaar niveau van 'aantal fouten in software' te behalen.

ad a) Het voorkomen van fouten

Het voorkomen van fouten wordt met name tijdens het ontwikkelen van de software toegepast. Een veelgebruikte aanpak hierbij is het concept van formele methoden, dat wil zeggen volgens vastgelegde procedures.

Een ander voorbeeld om het optreden van fouten te voorkomen is het gebruik van software ontwikkeld door verschillende teams (gebaseerd op identieke specificaties).

ad b) Het verwijderen van fouten

Om fouten te verwijderen worden testen uitgevoerd. De testmethoden proberen door het opstellen van uitgebreide en gedetailleerde testprotocollen uitputtend te testen of de software de beoogde functies juist uitvoert. Hierbij spelen een aantal problemen een rol.

Het eerste probleem is dat het praktisch onmogelijk is software uitputtend te testen. Door de toegenomen complexiteit van de gebruikte software kan men niet alle

---

1. Zie voor een behandeling van beoordelingsmethoden software bijvoorbeeld [BAE99, YOZ99, HAL00, DAP02, LIT02].
2. Zie voor een onderverdeling van methoden bijvoorbeeld [LEV95, STO96].

mogelijke responsen op alle mogelijke ingangssignalen testen om te beoordelen of er problemen zouden kunnen optreden.

Een veel gebruikte stelregel is dat er zelfs na zeer uitgebreide en diepgaande testen nog circa 10 fouten per 1000 regels software code aanwezig zijn.

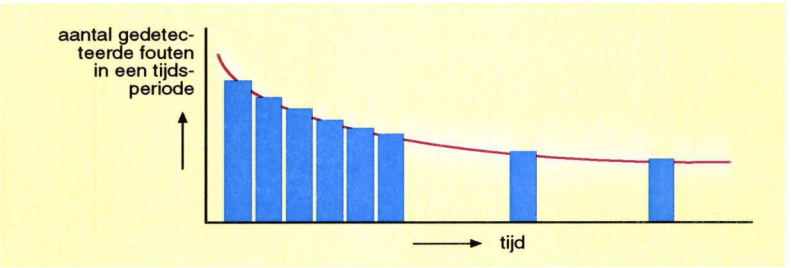
Het vinden van een fout

In een softwareprogramma voor het berekenenen van zwaar gas dispersie (bijvoorbeeld het vrijkomen van een wolk chloor of ammoniak) vermoedde men een fout. Een student heeft een half jaar naar de mogelijke fout gezocht en gevonden!. Hier gaat het over één fout waar men het vermoeden van heeft dat die aanwezig is, maar het zal een bijna onmogelijke opgave zijn om een complex programma geheel foutloos te krijgen.

Het tweede probleem is dat niet alleen zou getest moeten worden of de software de beoogde functie uitvoert maar ook dat er geen ongewenste functies worden uitgevoerd. Met andere woorden: de software moet doen wat deze moet doen, maar de software moet NIET doen wat deze NIET moet doen. Het testen hierop is zeer moeilijk.

Ook is het lastig om na te gaan of een bepaalde fout in de software daadwerkelijk problematische gevolgen heeft voor de beoogde werking van het *safety instrumented system*, hetzij onmiddellijk hetzij ten tijde van een potentieel gevaarlijke situatie.
Direct hieraan gerelateerd is het eerder geïntroduceerde concept van systematische fouten. Vaak werden software fouten gecategoriseerd onder systematische fouten. Er zijn enkele kwantitatieve methodes bekend om systematische fouten te modelleren. Hierbij wordt verondersteld dat de systematische fout vanaf ingebruikname van het SIS aanwezig is, maar pas 'tevoorschijn komt' wanneer bepaalde ingangssignalen aan het SIS worden toegevoerd.

ad c) Het detecteren van fouten
Het detecteren van fouten wordt gedaan tijdens het in bedrijf zijn van het SIS inclusief de software. Hierdoor worden mogelijke problemen door fouten in software voorkomen of de gevolgen ervan verminderd. Een praktische benadering is het aantal fouten dat wordt vastgesteld in een bepaalde periode in een diagram uit te zetten. Het aantal fouten in een periode moet dan na verloop van tijd gaan afnemen tot een constante waarde. Hiermee is het dan mogelijk om een voorspelling te doen wat het minimum aantal fouten zal zijn per tijdseenheid.



Figuur 5.4 Het verloop van het aantal gedetecteerde fouten in de tijd.

Ad d) Fouttolerantie

Het principe van fouttoleratie zorgt ervoor dat het SIS correct kan blijven functioneren, terwijl een fout in de software is opgetreden.

Aanbevolen technieken en standaarden

In de IEC 61508 standaard staan uitgebreide lijsten (tabellen) met in meer of mindere mate aanbevolen methoden en technieken ten behoeve van het vaststellen van de betrouwbaarheid van software. Deze methoden/technieken zijn gegeven per fase in het software ontwikkelproces (V-model, zie ook figuur 7.4). Voor een aantal methoden technieken zoals dynamische analyse en foutenanalyse zijn gedetailleerdere tabellen beschikbaar.

6. *Eisen aan SIS*

De belangrijkste eis te stellen aan een SIS is dat het bij een uit de hand gelopen proces een zodanige actie onderneemt dat het proces wordt teruggebracht naar een veilige toestand. Dit wordt ook wel functionele veiligheid (*functional safety*) genoemd. Daarnaast dient het SIS bedrijfszeker te zijn. Oftewel; het SIS dient zijn veiligheidsfuncties uit te voeren onder gedefinieerde omstandigheden gedurende een zekere tijd. Deze eigenschap wordt *safety integrity* genoemd en bepaalt mede welke risicoreductie in een proces te behalen is.

In dit hoofdstuk wordt eerst op het begrip safety integrity ingegaan en vervolgens op de hardware en software eisen te stellen aan het SIS. Bij het noemen van de eisen is uitgegaan van de Europese standaard IEC 61508[1].

6.1 Safety Integrity

Safety integrity en SIL klassen

De safety integrity of de bedrijfszekerheid van het SIS wordt gekarakteriseerd met de term 'Safety Integrity Level' of gewoon kortweg 'SIL'. In de standaard IEC 61508 zijn een viertal klassen gedefinieerd aangeduid met de nummers 1, 2, 3 en 4. Des te hoger het getal des te hoger de bedrijfszekerheid moet zijn van het SIS. Het SIL-nummer hangt nauw samen met de waarschijnlijkheid van falen van het SIS op aanspraak, dat wil zeggen als een ingreep door het SIS moet worden gedaan mag er maar een kleine (gemiddelde) kans zijn dat de actie niet wordt uitgevoerd. De waarschijnlijkheid van falen op aanspraak wordt algemeen aangeduid als *Probability of Failure on Demand* en wordt afgekort met PFD.

In onderstaande tabel zijn de kansen op falen bij aanspraak (PFD) weergeven voor de verschillende (SIL klassen). Hierbij is een onderscheid gemaakt tussen hoe vaak een SIS in een proces zal moeten ingrijpen. Dat kan zeer laag zijn, bijvoorbeeld minder dan een keer per jaar, of zeer regelmatig. In het laatste geval is wel sprake van een wat onrustig proces. Let wel dat in de tabel de waarschijnlijkheid voor hoge aanspraak per uur wordt uitgedrukt.

Tabel 6.1 Relatie tussen SIL en kans op falen van een SIS,

SIL	Waarschijnlijkheid van falen bij lage aanspraak (Probability of failure on demand, PFD)	Waarschijnlijkheid van falen per uur bij hoge aanspraak (Probability of a dangerous failure per hour)
4	$\geq 10^{-5}$ tot $< 10^{-4}$	$\geq 10^{-9}$ tot $< 10^{-8}$
3	$\geq 10^{-4}$ tot $< 10^{-3}$	$\geq 10^{-8}$ tot $< 10^{-7}$
2	$\geq 10^{-3}$ tot $< 10^{-2}$	$\geq 10^{-7}$ tot $< 10^{-6}$
1	$\geq 10^{-2}$ tot $< 10^{-1}$	$\geq 10^{-6}$ tot $< 10^{-5}$

SIL indeling volgens DIN en ANSI

Het Duitse instituut voor normering (DIN) heeft ook SIL klassen gedefinieerd in de TÜV standaarden[2] . Men noemt deze "*Anforderungsklassen*". Tabellen zijn beschikbaar voor het vertalen van SIL naar de "Anforderungsklassen"

1. Europese standard IEC 61508 [IEC001].
2. TÜV-standaarden: [DIN19251, DIN V 19250, DIN VDE 080].

Het Amerikaanse instituut voor standaardisatie ANSI (American National Standards Institution) heeft SIL-klassen gedefinieerd in de standaard ISA-S84.01[1]. In die standaard definieert men slechts 3 SIL-klassen, die overeenkomen met de eerste drie van de IEC 61508 standaard.

Noodzakelijke risicoreductie

Het SIS zit in de keten (operator, SIS, fysische beveiliging) die de uiteindelijke kans bepalen dat een ongewenste gebeurtenis bepalen. Des te lager de kans dat deze gebeurtenis zal optreden, des te groter de risicoreductie. Het SIS kan derhalve een belangrijke bijdrage leveren in de uiteindelijk te bereiken risicoreductie.

Het zal overigens niet in alle gevallen noodzakelijk zijn dat het hoogste niveau wordt geïnstalleerd. SIL1 geeft al een heel behoorlijke risicoreductie, SIL2 en 3 uiteraard nog beter, maar het is de vraag of dat nodig is. SIL 4 wordt niet toegepast in (chemische) processen alleen in systemen met de allerhoogste eisen aan betrouwbaarheid, bijvoorbeeld in de ruimtevaart.

SIL per veiligheidsfunctie

Opgemerkt moet worden dat een SIL geldt voor een specifieke veiligheidsfunctie, bijvoorbeeld het beveiligen op een zekere temperatuur kan een SIL2 nodig maken en het beveiligen op druk niet meer dan een SIL1. SIL moet dus per veiligheidsfunctie worden vastgesteld. Hiervoor geldt wel dat de veiligheidsfuncties geen onderlinge afhankelijkheden mogen hebben, anders dient dat als één veiligheidsfunctie gezien te worden.

Beschikbaarheid van het proces

Een SIS is bedoeld voor het beveiligen van een proces, maar het mag de beschikbaarheid van het proces niet aantasten door bijvoorbeeld zonder enige noodzaak een ingreep te doen die het proces stil (een "false-trip").

Deze laatste eis is meer een economische dan een functionele eis. Hoewel hier aan toegevoegd moet worden dat het opstarten en weer operationeel maken van een proces, na bijvoorbeeld een stillegging door het SIS, mogelijk ook meer risico's met zich meebrengt. Dit omdat dan allerlei zelden gebruikte procedures doorlopen moeten worden onder bijzondere procesomstandigheden, waarbij fouten kunnen worden gemaakt.

6.2 Hardware eisen in IEC 61508 standaard

Safe failure fraction

Hiervoor is aangegeven met welk SIL een zekere risicoreductie in een proces is te behalen. Dat kan dan zo wel gedefinieerd zijn, maar de apparatuur (de hardware) moet daar wel aan kunnen voldoen. Men heeft daarvoor twee begrippen geïntroduceerd safe *failure fraction* (de fractie van veilige fouten) en *hardware fault tolerance* (tolerantie voor hardware fouten). Daarnaast wordt voor het SIS onderscheid gemaakt in A en B systemen.

Safe failure fraction heeft te maken met situaties waarbij het systeem niet precies doet wat daarvan wordt verwacht, dus een fout in het meten en regelen. Men onderscheidt veilige fouten en gevaarlijke fouten. Veilige fouten hebben geen direct gevolg voor het proces en kunnen al dan niet worden gedetecteerd of gemeld. Gevaarlijke fouten zijn fouten als het SIS geen ingreep zal doen, maar als gezien de procesomstandigheden wel had gemoeten. Gevaarlijke fouten worden echter wel als veilig bestempeld als deze kunnen worden gedetecteerd. Het totaal aan veilige

1. [ISA96a],

fouten en gevaarlijke fouten, mits gedetecteerd, ten opzichte van het totaal aan mogelijke fouten bepaalt de safe failure fraction.

Voorbeeld veilige en gevaarlijke fouten

- *Stel we hebben een hydraulisch systeem met een druksensor. Als de druksensor faalt dan kan het systeem nog zijn functie uitvoeren. Het falen van de sensor kan worden gedetecteerd of niet, maar dat maakt voor de werking niet direct wat uit. Dit is een veilige fout.*
- *Een gevaarlijke fout is als het hydraulisch systeem druk verliest. Als dat wordt gedetecteerd dan is het nog een veilige fout, wordt het niet gedetecteerd dan is het een gevaarlijke fout.*

Hardware fault tolerance

De *hardware fault tolerance* is het maximum aantal (hardware) fouten dat aanwezig kan zijn in het systeem zonder dat het systeem (gevaarlijk) faalt. Onderscheid wordt gemaakt in 0, 1, en 2 fouten.

Voorbeeld fouttolerantie

Een logic solver heeft twee onafhankelijke I/O kaarten. Indien in één van de kaarten kortsluiting optreedt kan het systeem nog functioneren. De sluiting zal normaal gesproken worden gemeld en de kaart dient op een geschikt moment te worden vervangen.

Het systeem staat één fout toe en heeft dus een fouttolerantie van 1.

A en B systemen

A en B systemen hebben niets te maken met kwaliteit echter wel met wat we van een systeem weten. Een subsysteem is type A wanneer:

- Alle faalvormen (op welke wijze(n) een component kan falen) van alle componenten in het subsysteem helder gedefinieerd zijn;
- Het gedrag van het subsysteem ten tijde van fouten volledig bepaald kan worden;
- Betrouwbare faaldata vanuit praktijk beschikbaar is, zodanig dat aangetoond kan worden dat de toegelaten faalkans behaald wordt.

Een subsysteem is type B wanneer niet aan alle bovenstaande eisen voldaan wordt. Juist door het gebrek aan kennis over een B-systeem liggen de eisen voor een dergelijk systeem hoger. Het systeem zal minder snel in een hogere SIL-klasse worden ingedeeld.

Indelingstabel A en B systemen

Hieronder is de tabel gegeven die de relatie weergeeft welke SIL-klasse haalbaar is, gegeven een zekere fractie veilige fouten (safe failure fraction) en het type systeem (A of B) en de tolerantie voor veilige fouten (0, 1 of 2 fouten).

Tabel 6.2 Indelingstabel naar fractie veilige fouten, fouttolerantie en type systeem

Safe failure fraction	Hardware fault tolerance					
	Type A subsystem			Type B subsystem		
	Tolerantie voor veilige fouten					
	0	1	2	0	1	2
Geen (<60%)	SIL1	SIL2	SIL3	--	SIL1	SIL2
Laag (60-90%)	SIL2	SIL3	SIL4	SIL1	SIL2	SIL3
Gemiddeld (90-99%)	SIL3	SIL4	SIL4	SIL2	SIL3	SIL4
Hoog (>99%)	SIL4	SIL4	SIL4	SIL3	SIL4	SIL4

Uit de tabel valt af te leiden dat als bijvoorbeeld het percentage veilige fouten minder is dan 60% en het is een type A systeem met slechts een tolerantie van 0 (nul) fouten dan bereikt men niet meer dan SIL1. Kan het systeem één fout hebben zonder dat het gevaarlijk gefaald is dan zal men al SIL2 halen.

Kijken we naar de type B-systemen dan zal bij een fouttolerantie van 0 niet eens een SIL-klasse gehaald kunnen worden. Men krijgt SIL1 als het systeem toestaat dat bij één hardwarefout het systeem niet gevaarlijk faalt.

6.3 Software eisen in IEC 61508 standaard

De hardware eisen die in de IEC 61508 standaard staan zijn kwantitatief. De software eisen zijn echter kwalitatief. Dit is omdat fouten/problemen in software niet gekwantificeerd kunnen worden.

De eisen die in de IEC 61508 staan geformuleerd met betrekking tot zowel embedded als application software van een SIS richten zich met name op het ontwerp- en ontwikkelproces van software. Deze eisen zijn daar waar mogelijk in lijn met de kwalitatieve eisen met betrekking tot hardware. Er wordt echter wel dieper ingegaan op specifieke software aspecten zoals gebruikte programmeertalen en interne testprocedures.

Er zijn verschillende technieken om te voldoen aan de software eisen die gesteld worden. Onderstaande tabel laat een aantal voorbeelden zien. Het voert hier te ver om op deze technieken in te gaan.

Tabel 6.3 Enkele aanbevolen technieken in de standaard IEC 61508 teneinde aan software eisen te kunnen voldoen.

Methode om fouten in software vast te stellen	SIL1	SIL2	SIL3	SIL4
1 Fault detection and diagnostics	---	R	HR	HR
2 Error detecting and correcting codes	R	R	R	HR
3a Failure assertion programming	R	R	R	HR
3b Safety bag techniques	---	R	R	R
...	...	...	...	...

R = Recommended (aanbevolen), HR = Highly recommended (zeer aanbevolen) .

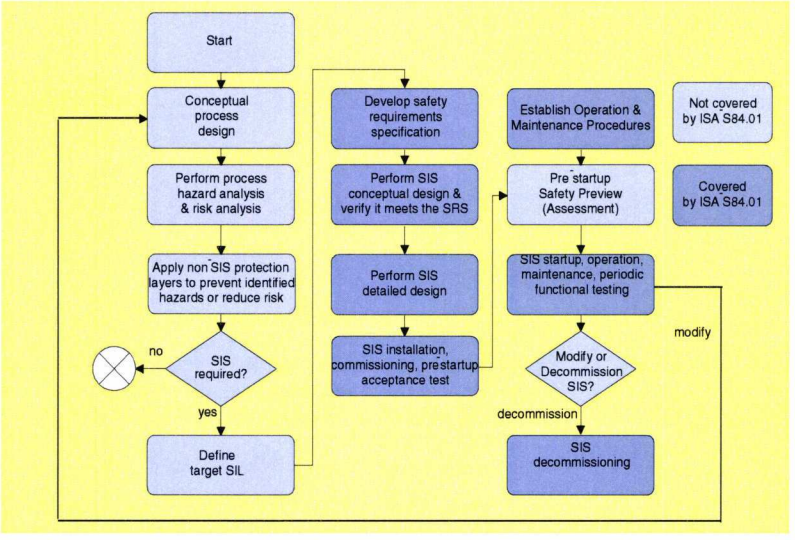
Om een hogere SIL-klasse te kunnen bereiken zal men meerdere technieken moeten toepassen om de betrouwbaarheid te verhogen.

7. Ontwerpproces volgens standaarden

Het SIS wordt meestal ontworpen voor een bepaalde toepassing of proces. De te nemen stappen in het ontwerpproces zijn in standaarden weergegeven. Hieronder worden de aanbevolen werkwijzen volgens de Amerikaanse standaard ISA-S84.01 en de Europese standaard IEC 61508 weergegeven. De stappen volgens Amerikaanse standaard wordt hier ook gegeven vanwege de vele overeenkomsten tussen de Europese standaard en omdat een SIS ook op de Amerikaanse standaard kan zijn gebaseerd.

7.1 Te nemen stappen volgens ISA-S84.01 standaard

Het stappenschema volgens de Amerikaanse ANSI-standaard is in onderstaande figuur weergegeven. In de standaard worden de fasen met betrekking tot de *safety requirements specification*, ontwerp en ontwikkeling, ingebruikname, onderhoud en ontmanteling volgens het lifecycle principe uitgebreid behandeld. De fasen zijn in de figuur donkerblauw gekleurd aangegeven. De overige stappen worden niet behandeld in de standaard, maar zijn wel bepalend voor de uiteindelijke uitvoeringsvormen.



Figuur 7.1 Safety lifecycle zoals gedefinieerd door ANSI/ISA-S84.01 standaard.

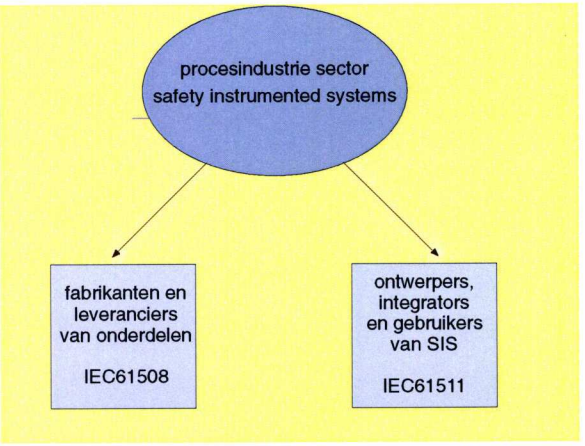
Uit het schema is af te lezen dat de start is het conceptuele ontwerpproces. Vervolgens moet in ieder geval een gevaren- en een risicoanalyse moet worden uitgevoerd. Deze zal moeten worden herhaald als er een modificatie in de SIS wordt aangebracht. Na de risicoanalyse wordt de vraag gesteld of een SIS nodig is, zo ja dan dient het te behalen SIL (target SIL) te worden gedefinieerd. Voorafgaande daaraan dient eerst nog nagegaan te worden of met andere middelen een risicoreductie is te behalen.

7.2 Te nemen stappen volgens de IEC 61508 standaard

Status en inhoud standaard

De Europese IEC 61508 standaard is de meest recente en geaccepteerde standaard op het gebied van *safety instrumented systems*. Deze op risico gebaseerde kapstok- of paraplu standaard is ontwikkeld met als voornaamste doel de functionele veiligheid van zogenaamde E/E/PE safety-related systems[1] te waarborgen. Daarnaast wordt sinds enige tijd de IEC 61511 standaard ontwikkeld als zijnde de applicatie specifieke implementatie van de IEC 61508 voor de procesindustrie.

De IEC 61508 standaard is bedoeld voor ontwikkelaars van SIS componenten en subsystemen, terwijl de IEC 61511 (concept) standaard bedoeld is voor gebruikers en integrators van SIS's, zie figuur 7.2

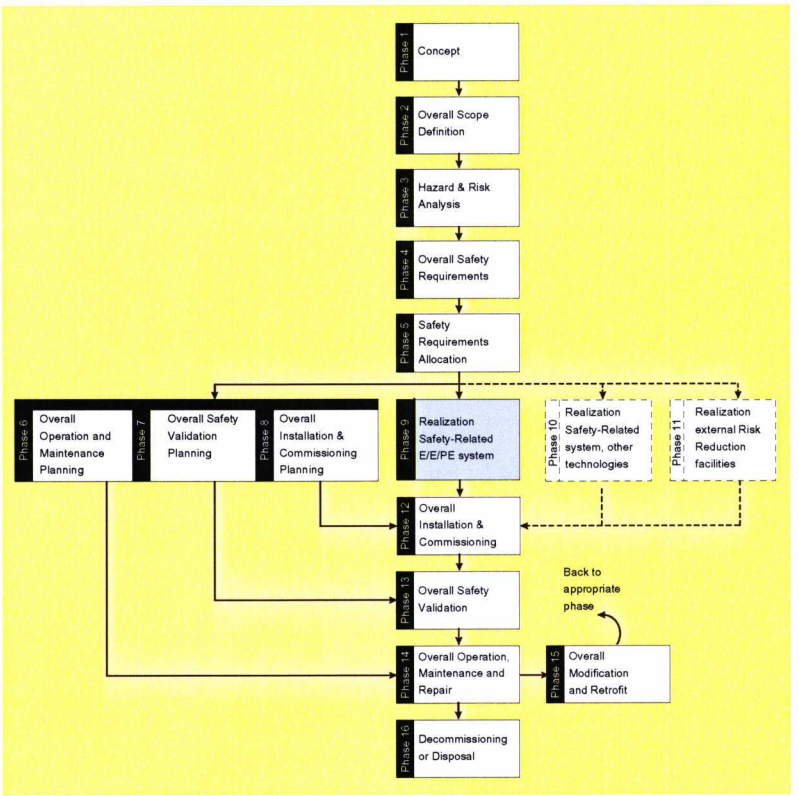


Figuur 7.2 Onderscheid gebruik IEC 61508 standaard en IEC 61511 (concept) standaard.

Stappen in de Overall Safety Life Cycle

De IEC 61508 definieert een aantal fasen voor de ontwikkeling, ingebruikname, onderhoud en ontmanteling van *E/E/PE safety-related systems*. Dit als technisch raamwerk om op een systematische wijze te kunnen omgaan met alle activiteiten die nodig zijn om de functionele veiligheid en de betrouwbaarheid te kunnen garanderen van *E/E/PE safety-related systems*. De fasen volgens IEC 61508 worden schematisch weergegeven in figuur 7.3.

1. E/E/PE = Electrical/Electronic/Programmable Electronic.



Figuur 7.3 Fasen zoals gedefnieerd in IEC 61508.

Men ziet in de figuur een aantal aandachtspunten terugkomen die ook in de Amerikaanse standaard worden genoemd, zoals een gevarenanalyse, die eventueel herhaald moet worden bij modificatie. Verder is veel aandacht voor planning.

Kwalitatieve eisen in het ontwikkelingsproces

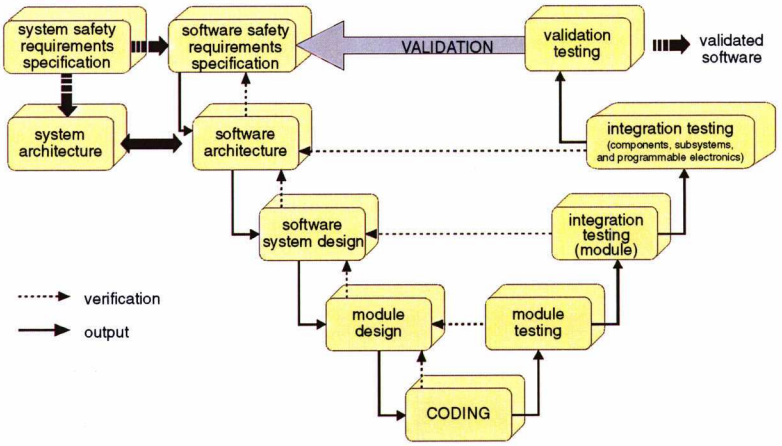
In de standaard IEC 61508 worden naast de hardware eisen ook kwalitatieve eisen genoemd voor het ontwikkelingsproces. Per fase worden de volgende aspecten besproken:

- Vereiste invoerinformatie
- Doelstelling van de fase, inclusief scope
- Te vervullen eisen
- Te produceren output (documentatie)

Het laatste aandachtspunt documentatie is een van de pijlers waarop het ontwikkelingsproces kan worden beoordeeld.

Software safety lifecycle (het V-model)

Behalve de fasen voor hardware zoals getoond in figuur 7.3 moeten ook een aantal stappen worden doorlopen voor de ontwikkeling van software. Men hanteert daarvoor het zogenaamde V-model (zie figuur 7.4).



Figuur 7.4 V-model zoals gehanteerd door IEC 61508 standaard.

Zoals in de figuur is te zien dienen eerst de *system safety requirements* (veiligheidseisen) te worden gespecificeerd. Dit moet weer worden vertaald in software safety requirements specificaties. Beiden bepalen de opbouw van het systeem en de software. Daarna volgen de stappen betreffende het algemene ontwerp en het module ontwerp.

Modules kunnen speciaal geschreven kleine pakketten software of bibliotheekprogramma's zijn (de vorm waarin de modules te verkrijgen zijn is besproken in 4.3.2). Hierna volgt het eigenlijke programmeren of het aan elkaar schrijven van het programma en daarna de verschillende teststappen.

Verficatie en validatie

Voor een juist begrip dient nog het verschil tussen verificatie en validatie te worden genoemd. De verificatie activiteit zoals weergegeven in figuur 7.4 betreft de beoordeling of het ontworpen systeem (software) **goed** ontworpen is. Het valideren van het ontworpen systeem betreft het beoordelen of het **juiste** systeem is ontworpen, derhalve geschikt voor zijn beoogde veiligheidsfunctie.

8. Checklists

De ontwikkelaars van een SIS moet een groot aantal stappen zetten om tot een betrouwbaar eindproduct te komen. Dat geldt voor de hardware maar ook voor de software. Zij zullen voor hun werk diverse checklists hanteren. Een checklist van geheel andere aard is de checklist om na te gaan of wel alle stappen zijn gedaan zodat men vertrouwen kan hebben in het geleverde product.

Hieronder wordt toegelicht **wat** van het uitgevoerde ontwikkelproces moet worden gecheckt en **wie** dat moet doen . Een tweetal verkorte checklists worden gegeven in bijlage 4. De checklists gaan in op de hoofdpunten. De eerste checklist gaat voornamelijk of de noodzakelijke documentatie aanwezig is en de tweede checklist over onafhankelijkheid bij het checken. Beide checklists zijn opgesteld door TNO in het kader wat gewenst is als instrumenten voor arbeidsinspecteurs.

8.1 Check op documentatie

Waarom checken op documentatie?

Elke fase in van het ontwikkelingsproces moet volgens de IEC 61508 worden gedocumenteerd. Hierin is dan ook alle informatie te vinden die van belang is voor een beoordeling van het ontwikkelingsproces.
Het doorgraven van de documentatie geeft inzicht over het wie, wat en hoe van het ontwikkelinsproces. Ten aanzien van wie, wat en hoe wordt opgemerkt dat verschillende functionarissen, afdelingen en ook bedrijven een bijdrage aan het ontwerp zullen hebben gegeven. Zo wordt een eerste beschrijving van het te beveiligen proces door een procestechnoloog gedaan, al dan niet ondersteund door laboratorium- en pilotproeven.
De process engineer moet zorgen voor de het feitelijke ontwerp met alle "ins en outs". De veiligheidsman dient er zorg voor te dragen dat het veiligheidsplan correct is en dat een hazard- en risicoanalyse wordt uitgevoerd en aanbevelingen worden geïmplementeerd.
Het management dient er op toe te zien dat de procedures worden gevolgd. De leverancier van SIS moet zich verdiepen in het proces en de softwareman moet op basis van verstrekte informatie een programma schrijven. Gezien de vele betrokkenen en stappen in informatieoverdracht is het nodig dat voor het gehele ontwerp een onafhankelijke toets wordt gedaan, waarbij de gehele keten wordt beoordeeld.

Volledigheid documentatie

Een eerste check op de toegedachte functie van een SIS ligt derhalve in het vaststellen of de documentatie aanwezig is. Vervolgens kan worden nagegaan of de documentatie volledig en up-to-date is. Verder zal het duidelijk moeten zijn dat het SIS ook volgens de vastgelegde procedures is ontworpen en zal worden onderhouden. In checklist 1 in bijlage 4 kunnen de bevindingen worden weergegeven.

Welke documentatie moet aanwezig zijn?

Een ontwerp van een SIS is gebaseerd op een aantal uitgangspunten. Deze zullen in een aantal documenten zijn vastgelegd. Deze zijn:

- Een veiligheidsrapport, waarin is aangegeven wat de gevaarsaspecten zijn van het proces.
- Een functionele specificatie of procesbeschrijving, waarin is weergegeven de waarden van de procesvariabelen en kritische waarden.

- Een proces- en instrumentatiediagram (P&ID), waarin duidelijk de processtromen en de meetpunten zijn weergegeven.
- Rapport van een hazard- en risicoanalyse. Deze analyse moet zijn uitgevoerd om de mogelijke afwijkingen van het proces in beeld te brengen en tevens of voldoende maatregelen zijn genomen ter beperking van het risico. Met behulp van de hazard- en risicoanalyse moet ook het benodigde SIL per uit te voeren veiligheidsfunctie worden vastgesteld.
- Een safety plan met logische diagrammen. Het plan geeft aan op welke wijze het proces wordt beheerst met te installeren SIS's.

Verder is van belang dat het ontwerp van een SIS wordt uitgevoerd volgens procedures vastgelegd in:
- Een kwaliteitsplan
- Configuratie management procedures. In deze procedures moet zijn vastgelegd hoe veranderingen in een SIS worden gedaan en worden gedocumenteerd.

8.2 Door wie moet worden gecheckt?

De vraag door wie moet worden gecheckt hangt deels samen met twee invalshoeken: de vakbekwaamheid van een beoordelaar en zijn onafhankelijkheid. Voor beide aspecten zijn richtlijnen gegeven.

Vakbekwaamheid

Het checken van het beschreven ontwerp van een SIS moet gebeuren door een beoordelaar (assessor) met een zekere vakbekwaamheid. De IEC 61508 standaard stelt dat een beoordelaar het juiste niveau van vakbekwaamheid moet hebben ten aanzien van:
- Het van toepassing zijnde terrein.
- De van toepassing zijnde technologie (bijvoorbeeld op electrisch, electronisch en software gebied).
- Safety engineering met betrekking tot de technologie waar het SIS zijn toepassing vindt.
- De van toepassing zijnde veiligheidsvoorschriften.
- De training, ervaring en kwalificaties van een beoordelaar moeten gedocumenteerd zijn.

Onafhankelijkheid

Men onderscheidt voor onafhankelijkheid een drietal mogelijkheden. Het checken wordt gedaan door een onafhankelijke persoon, onafhankelijke afdeling of een onafhankelijke organisatie. Onafhankelijk wil zeggen niet betrokken bij het ontwerpproces.

De eis van onafhankelijkheid wordt zwaarder naarmate het SIL van een SIS hoger is. Tevens dient daarbij rekening te worden gehouden met:
- Complexiteit en nieuwheid van het ontwerp.
- Onbekendheid van de toe te passen technologie en hierdoor gebrek aan ervaring met het ontwerp.
- Minder standaardisatie in het ontwerp.

Tabel relatie onafhankelijkheid en SIL

De aanbevelingen wie bij voorkeur de onafhankelijke beoordeling moet doen in relatie tot het SIL zijn in tabel 8.1 weergegeven.

Tabel 8.1 Onafhankelijkheid beoordelaar afhankelijk van SIL.

Minimum level of Independence	Safety integrity level			
	1	2	3	4
Independent person	HR	HR^1	NR	NR
Independent department	-	HR^2	HR^1	NR
Independent organization	-	-	HR^2	HR

Independent: Onafhankelijk

NR: Not Recommended: niet aanbevolen (afgeraden).

HR: Highly Recommended: sterk aanbevolen.

HR^1 of HR^2: voor een onderscheid in hiervoor genoemde zaken wordt de aanbeveling HR (Highly Recommended) verder ingedeeld in HR^1 en HR^2. Er wordt gebruik gemaakt van aanbeveling HR^2 wanneer de eerder genoemde zaken van toepassing zijn te weten complexheid, onbekendheid en minder standaardisatie.

Opmerking: In de standaard IEC 61508 worden meer details gegeven over hoe de bovenstaande tabel moet worden geïnterpreteerd.

Uit de tabel volgt dat de mate van gewenste onafhankelijkheid van de beoordeling met de SIL-klasse toeneemt zoals ook te verwachten is. Verder valt uit de tabel af te lezen dat de beoordelaar bij SIL2 een onafhankelijk persoon kan zijn (HR^1). Zijn de zaken niet zo duidelijk dan zal de beoordeling moeten worden gedaan door een onafhankelijke afdeling (HR^2).

Checklist beoordelaars

De beoordelaars kunnen vervolgens ook weer worden beoordeeld of zij de beoordeling in volledigheid hebben uitgevoerd. In checklist 2 (bijlage 4) kunnen de bevindingen worden weergegeven.

De ervaring van de beoordelaar(s) is een belangrijk gegeven, zeker bij een hogere SIL. De beoordelaars van een onafhankelijke organisatie dienen bij voorkeur in het bezit te zijn van een certificaat waarin is aangegeven tot welk niveau zij mogen beoordelen.

9. Referenties

[AAR89] Aarø, R., Bodsberg, L., Hokstad, P., *Reliability Prediction Handbook; Computer-Based Process Safety Systems*, Trondheim, Foundation for Scientific and Industrial Research at the Norwegian Institute of Technology (SINTEF), 1989

[AND93] Andrews, J.D., Moss, T.R., *Reliability and Risk assessment*, Longman Scientific & Technical, 1993, ISBN / ISSN 0-582-09615-4 / 0-470-23345-1

[ASC00] Asschenbrenner, S., Houtermans, M., *IEC 61508 and Management of Functional Safety*, ISA/EXPO 2000, New Orleans, USA, August 2000

[ASH96] Ashton, P., Leicht, R., *Modelling Industrial Systems using Fault Tree and Monte Carlo Analysis Techniques*, Proceedings of the Probabilistic Safety Assessment and Management Conference (ESREL '96 - PSAM III), London, Springer-Verlag, 1996

[BAE99] Baer, D., Boyd M., Aschenbrenner, S., *Software Requirements to a SIL 2 Application According to IEC 61508*, proceedings ISA/TECH99 conference, Philadelphia, USA, 1999

[BAK99] van Bakel, F., Knegtering, B., *Experiences with Organizational Aspects of Implementing IEC 61508 Safety Standard into an Existing Quality Management System*, proceedings ISA/TECH99 conference, Philadelphia, USA, 1999

[BEU02] Beurden, I.J.W.R.J. van, Amkreutz, R., *Safety in Batch production*, Exida.com, 2002

[BOY95] Boyd, M.A., *What Markov Modeling Can Do for You: An Introduction*, Proceedings of Annual Reliability and Maintainability Symposium, New York, Institute of Electrical and Electronics Engineers (IEEE), 1995

[BRO99a] A.C. Brombacher, *Maturity index on reliability: covering non-technical aspects of IEC 61508 reliability certification*, Reliability Engineering & System Safety, Vol.66, No. 2, November 1999

[BRZ99] *Besluit risico's zware ongevallen 1999*, Artikel 5.2 en Staatsblad 234, 17 juni 1999

[BT87] *Handbook of Reliability Data for Components used in Telecommunications Systems, Issue 4*, British Telecom, 1987

[CCP01] *Layer of Protection Analysis: Simplified Process Risk Assessment*, Center for Chemical Process Safety, American Institute of Chemical Engineers, New York, ISBN 0-8169-0811-7, 2001

[CCP92] *Guidelines for Hazard Evaluation Procedures, Second Edition with Worked Examples*, Center for Chemical Process Safety, American Institute of Chemical Engineers, New York, ISBN 0-8169-0491-X, 1992

[CCP93] *Guidelines for Safe Automation of Chemical Processes*, NY: New York, Center for Chemical Process Safety, American Institute of Chemical Engineers, 1993.

[DAP02] Dapena, P., *Software safety verification in critical software intensive systems*, PhD thesis, Eindhoven University of Technology, March 2002, ISBN 90-386-0953-0

[DEF00] HAZOP *Studies on systems containing programmable electronics*, Ministry of Defence, Defence standard 00-58, Issue 2, 2000

[DEF97] *Requirements for Safety Related Software in Defence Equipment*, Ministry of Defence, Defence standard 00-55, Issue 2, 1997

[DIN90] DIN V VDE 0801/01.90, *Grundsätze für Rechner in Systemen mit Sicherheitsaufgaben*, Berlin, Deutsches Institut für Normung ev., 1990.

[DIN92] DIN 19251, *MSR Schutzeinrichtungen Anforderungen und Massnahmen zur gesicherten Funktion*, Berlin, Deutsches Institut für Normung ev., 1992

[DIN94A] DIN V 19250, *Grundlegende Sicherheitsbetrachtungen für MSR-Schutzeinrichtungen*, Berlin, Deutsches Institut für Normung ev., 1994.

[DIN94B] DIN V VDE 0801 A1, *Grundsätze für Rechner in Systemen mit Sicherheitsaufgaben, Änderung A1*, Berlin, Deutsches Institut für Normung ev.,1994.

[FLE74] Fleming, K.N., *A probabilistic model for common mode failures in redundant safety systems*, US: General atomic report CA-13284, 1974

[GOB92] Goble, W.M., *Evaluating Control Systems Reliability Techniques and Applications*, Instrument Society of America, 1992

[GOB98A] Goble, W.M., *The use and development of quantitative reliability and safety analysis in new product design*, Ph.D. thesis, Eindhoven University of Technology, Netherlands, 1998, ISBN 90-386-0870-5

[GOB98B] Goble, W.M., *Evaluating Control Systems Reliability Techniques and Applications*, 2nd edition, Instrument Society of America, 1998

[GOB99] Goble, W.M., Brombacher, A.C., *Using a failure modes, effects and diagnostic analysis (FMEDA) to measure diagnostic coverage in programmable electronic systems*, Reliability Engineering & System Safety, Vol. 66, No. 2, November 1999

[GRU98] Grühn, P., Cheddie, H., Safety shutdown systems, Design, Analysis, and Justification, 1998, ISBN 1556176651

[HAL00] Halang, W.A., *Programmable control on the Safety Integrity Levels 3 and 4*, TÜV 4th international symposium on programmable electronic systems in safety related applications, 2000, Cologne, Germany

[HSE01] *Justifying the use of software of uncertain pedigree (SOUP) in safety-related applications*, Health and Safety Executive contract research report 336/2001, 2001

[HSE92] *The tolerability of risk from nuclear power stations*, Health and Safety Executive (UK) publication, ISBN 011 886368 1, 1992

[HSE95] Health and Safety Executive, *Out of Control, Why control systems go wrong and how to prevent failure*, Health and Safety Executive, 1995, ISBN 0 7176 0847 6

[IEC00] IEC 61508, *Functional Safety of electrical/electronic/programmable electronic safety-related systems*, Genève, Bureau Central de la Commission Electrotechnique International, 2000

[IEC01] Draft standard IEC 61511, *Functional safety: safety instrumented systems for the process industry sector*, Genève, Bureau Central de la Commission Electrotechnique International, 2001

[IEC88] IEC 60812, *Analysis Techniques for system reliability – Procedure for failure mode and effects analysis*, Genève, Bureau Central de la Commission Electrotechnique International, 1988

[IEC90] IEC 1025, *Fault Tree Analysis* (FTA), Genève, Bureau Central de la Commission Electrotechnique International, 1990

[IEC91] IEC 1078, *Analysis Techniques for Dependability – Reliability Block Diagram Method*, Genève, Bureau Central de la Commission Electrotechnique International, 1991.

[IEC92] IEC 61131, *Programmable Controllers*, Genève, Bureau Central de la Commission Electrotechnique International, 1992

[ISA96A] ISA S84.01-1996, *Application of Safety Instrumented Systems for the Process Industries*, NC: Research Triangle Park, Instrument Society of America, 1996

[ISA96B] dTR84.0.02, draft Technical Report, *Electrical (E) / Electronic (E) Programmable electronic Systems (PES) - Safety Evaluation Techniques*, NC: Research Triangle Park, Instrument Society of America, 1996.

[KLE92] Kletz, T.A., *Hazop and Hazan: identifying and assessing process industry hazards*, Rugby: Institution of Chemical Engineers, ISBN/ISSN: 1-56032-276-4, 0-85295-285-6, 1992

[KLE94] Kletz, T.A., *What went wrong? Case histories of process plant disasters*, London: Gulf Publishing Company, 1994, ISBN 0-88415-027-5

[KNE02] Knegtering, B., *Safety lifecycle management in the process industries: the development of a qualitative safety related information analysis technique*, Eindhoven: Technische Universiteit Eindhoven, 2002, PhD thesis, ISBN 90-386-1747-X

[KUM96] Kumamoto, H., Henley, E.J., *Probabilistic Risk assessment and Management for Engineers and Scientists (second edition)*, New York, Institute of Electrical and Electronics Engineers (IEEE), 1996

[LEV95] Leveson, N.G., *Safeware: system safety and computers*, Amsterdam: Addison-Wesley, ISBN 0-201-11972-2, 1995

[LIT02] Littlewood, B., Popov, P., Strigini, L., *Assessing the reliability of diverse fault-tolerant software-based systems*, Safety Science, issue 40, p. 781-796, December 2002

[MIL84] US MIL-STD-1629A: *Procedures for Performing a Failure Mode Effects and Criticality Analysis*, National Technical Information Service, VA: Springfield, 1984

[MIL92] US MIL-HDBK-217F: *Military Handbook Reliability Prediction of Electronic Equipment*, National Technical Information Service, VA: Springfield, 1992

[NEU95] Neumann, P.G., *Computer-Related Risks*, Addison-Wesley, Association for Computing Machinery (ACM) Press, 1995, ISBN 0-201-55805-X

[ORE97] Oreda participants, *Offshore Reliability Data Handbook 3rd Edition*, 1997 (OREDA-97)

[PER84] Perrow, Ch., *Normal accidents: living with high-risk technologies*, New York Basic Books, 1984, ISBN 0-465-05144-8

[RIS02] *Risico analyse in Vogelvlucht*, to be published, 2002

[ROU97] Rouvroye, J.L., Houtermans, M.J.M., Brombacher, A.C., *Systematic Failures in Safety Systems: Some observations on the ISA-S84 standard*, Proceedings ISA TECH/EXPO volume 1, Anaheim, 1997

[ROU01] Rouvroye, J.L., *Enhanced Markov Analysis as a method to assess safety in the process industry*. PhD thesis, May 2001, ISBN 90-396-2772-6

[SEV96] Seveso II, *Directive 96/82/EC on the control of major-accident hazards*, Council of the European Union, 1996

[STO96] Storey, N., *Safety-critical computer systems*, Amsterdam: Addison-Wesley, ISBN 0-201-42787-7, 1996

[SUM99] Summers, A., *Estimation and evaluation of common cause failures in SIS*, Chemical Engineering Progress, November 1999

[VES81] Vesely, W.E., Goldberg, F.F., Roberts, N.H., Haasl, D.F., *Fault Tree Handbook*, U.S. Nuclear Regulatory Commission, NUREG-0492, Springfield, 1981

[WAS75] *Reactor Safety Study*, WASH-1400 (NUREG-75/014), United States Nuclear Regulatory Commission, October 1975

[WAT61] Watson, H.A., Bell Telephone Laboratories, *Launch Control Safety Study*, Bell Telephone Laboratories, Murray Hill, USA

[WRA96] *Common-Cause Failures in Relation To Programmable Electronic Systems Used for Protection*, A. M. Wray, Health and Safety Laboratory, Report to IEC 1508 Committee, August 1996

[YOZ99] Yozallinas, J.C., Goble, W.M., *Software Safety & Reliability Using IEC 61508 Techniques*, proceedings ISA/TECH99 conference, Philadelphia, USA, 1999

10. *Afkortingen*

In de hoofdtekst en in de bijlagen van dit boekje worden veel afkortingen gebezigd. In de onderstaande tabel staan de afkortingen en hun betekenis weergegeven.

Afkorting	Betekenis
CC	Common cause (gemeenschappelijke oorzaak)
COTS	Commercial off-the-shelf software (commercieel verkrijgbare programma's)
D	Diagnostics
DIN	Deutsches Institut für Normung e.V.
EUC	Equipment under control
E/E/PE	Electrical / electronic / programmable electronic
FLD	Functional Logic Diagram
FMECA	Failure Mode Effects and Criticality Analysis
FTA	Fault Tree Analysis
HazOp	Hazard and Operability analysis
HR	Highly recommended
IEC	International Electrotechnical Commission
ISA	Instrumentation, Systems, and Automation society (voorheen: Instrument Society of America)
LOPA	Layer of Protection Analysis
MooN	M-out-of-N
PFD	Probability of Failure on Demand
PFDavg	Average Probability of Failure on Demand
P & ID	Piping and Instrumentation Diagram
PLC	Programmable Logic Controller
SIF	Safety Instrumented Function
SIL	Safety Integrity Level
SIS	Safety Instrumented System
SOUP	Software of unknown pedigree
SRS	Safety-related system
SRS	Safety requirements specification

Bijlage 1 Lijst van softwaregerelateerde ongelukken

Bestaande uit: artikel Volkskrant en Microsoft Executive e-mail (Engels)

VOLKSKRANT ARTIKEL

In de Volkskrant van zaterdag 23 november 1996 stond een artikel met een overzicht van enkele bekende ongelukken veroorzaakt door software. Dit artikel staat hieronder weergegeven.

Volkskrant: Uit het katern WETENSCHAP van ZATERDAG 23 NOVEMBER 1996

Alleen opletten helpt tegen fatale software

Organisaties hebben het vaak moeilijk als er een nieuw computersysteem wordt ingevoerd. Hoe goed de fabrikant ook test, veel programmafouten komen pas in de praktijk aan het licht. De gevolgen kunnen desastreus zijn.

DE FOUT WERD vorige maand ontdekt en bezorgt programmeurs nog steeds overuren. Iedere bezitter van een pc met Windows 95 en een Internetaansluiting hoeft maar één regel in te tikken en een computer elders in de wereld valt uit, bijvoorbeeld de pc van een nietsvermoedende Internet-gebruiker. De Ping of Death, zoals de fout heet, heeft weer toegeslagen. Aanstichter van dit alles is het programma Ping. Alle op Internet aangesloten computers versturen met Ping kleine pakketjes naar elkaar om te melden dat ze er nog zijn. Het is een routineprocedure die dagelijks miljoenen malen wordt herhaald.

De versie van Ping die in Windows 95 zit en in enkele andere besturingssystemen, kan ook pakketten verzenden die enkele bytes groter zijn dan het gebruikelijke maximum. Als de ontvangende computer die extra bytes niet afvangt, komen ze soms in een vitaal deel van het computergeheugen terecht. Het gevolg is dat het systeem er de brui aan geeft. Windows 95 zelf is nauwelijks gevoelig voor te grote pakketten, maar veel andere besturingssystemen voor pc's, Apples en grotere machines zijn dat wel. Programmeerfouten zijn er in alle soorten en maten, van irritant tot levensbedreigend. De Ping of Death valt in de eerste categorie en is een typisch voorbeeld van hoe fouten ontstaan in computernetwerken die bestaan uit verschillende systemen. De makers van de diverse programma's met een versie van Ping erin hielden zich aan een conventie: de maximum grootte van een pakketje. De interactie tussen die verschillende Pings leidde daardoor niet tot ongelukken. Het ging pas mis toen Windows 95 zich niet meer aan de conventie hield. Dat was iets waar de programmeurs van oudere systemen geen rekening mee hadden gehouden. Anderzijds zou zelfs met de meest uitgebreide praktijktest de fout niet zijn opgespoord, simpelweg doordat de fatale omstandigheid pas door Windows 95 werd gecreëerd. Systemen kunnen perfect werken bij introductie, maar verborgen gebreken hebben die pas onder nieuwe omstandigheden aan het licht komen. Een klein computerongemak is nog niet zo'n ramp. Maar het kan erger. Op 11 april 1986 lag de 66-jarige Vernon Kidd onder het bestralingsapparaat Therac 25 in het East Texas Cancer Center om de huidkanker aan de rechterkant van zijn gezicht te laten behandelen. Maar in plaats van het gebruikelijke niks voelde hij deze keer een brandende pijn.

Drie weken later overleed hij aan een hersenbeschadiging ten gevolge van een overdosis straling. Uit het onderzoek dat volgde naar aanleiding van zes ernstige incidenten, waaronder drie dodelijke, bleek dat de software van de Therac 25 een rommeltje was, een slordige collage van nieuwe code en meer dan tien jaar oude stukjes uit de programma's van voorgangers. Telkens dacht de fabrikant de fout gevonden te hebben, maar iedere keer dook een nieuwe fout op. Zo bleek de volgorde waarin de behandelinstelling werd ingetikt van invloed te zijn op de timing van het apparaat.

DERGELIJKE FOUTEN kunnen binnensluipen als gevolg van minieme slippertjes van de programmeur. Als een ingetypt stuk informatie moet worden veranderd, roept het hoofdprogramma een deelprogramma aan dat de cursor op de juiste plek zet, de nieuwe informatie van het toetsenbord leest en die op het scherm weergeeft. Maar de nieuwe gegevens moeten ook expliciet worden doorgegeven aan het hoofdprogramma, anders handhaaft dat de oude informatie. Er zijn immers ook gevallen waarbij een deelprogramma juist niet aan het hoofdprogramma mag morrelen. Vaak is het echter zo logisch dat het hoofdprogramma de nieuwe informatie ook moet hebben, dat een slordige programmeur vergeet een expliciete opdracht tot verandering te geven. De fouten in de Therac 25 zouden bij een degelijke testprocedure zeker zijn achterhaald. Bij een goede test krijgt het programma een grote hoeveelheid verschillende invoer te verwerken, waarbij de testers erop letten dat alle delen van het programma aan bod komen, ook die welke geacht worden uitzonderingen op te vangen. Een deelprogramma dat systematisch dezelfde fout maakt, is zo makkelijk te ontdekken, in tegenstelling tot een deelprogramma dat, zoals Ping, alleen onder zeer specifieke omstandigheden de mist in gaat. Een programma van een beetje omvang kent namelijk zoveel invoercombinaties dat die onmogelijk allemaal te testen zijn. De praktijk is zodoende onvermijdelijk de enige ultieme test. De gebruikers slagen er altijd wel in omstandigheden te vinden waar de ontwerpers van de software niet aan hebben gedacht.Eerder dit jaar deed zich bijvoorbeeld een incident voor met een Airbus 320 van Northwest Airlines, een toestel dat electronisch wordt bestuurd. Tijdens de daling naar het vliegveld van Washington DC maakte het toestel,plotseling een draai van dertig graden waar de piloot niet om gevraagd had.

Het systeem bleek bij erg intensieve manoeuvres, zoals het dalen, gevoelig te zijn voor overbelasting. De computer kreeg dan zoveel opdrachten tegelijk dat hij die niet meer allemaal aankon en sommige in de wachtkamer zette. De piloot merkte dat er te weinig gebeurde en corrigeerde totdat hij wel op de juiste koers zat. Op dat moment zaten er echter nog een heleboel correcties in de wachtkamer en toen die ook doorgevoerd werden schoot het vliegtuig te ver door.

Dit incident liep goed af, net als een tiental gelijksoortige. Maar drie van deze toestellen crashten, waaronder de Airbus die op 20 januari 1992 in de Vogezen neerkwam. Van de 96 inzittenden kwamen er 87 om. Bij crashes als deze luidde het oordeel steeds: fout van de piloot.

VLIEGTUIG-BOUWERS en makers van andere kritieke systemen (kerncentrales, space shuttles, enzovoorts) getroosten zich veel moeite om met strenge procedures, voortdurende dubbelchecks en uitgebreide tests het aantal fouten tot een minimum te beperken. Boeing gooide met het oog hierop het hele ontwerpproces van de 777 om. Ondanks alle voorzorgen bleek echter al snel na de introductie dat de boordcomputer de gevolgen van windstoten overcompenseerde waardoor de staart onnodig heen en weer zwaaide. Makers van consumentensoftware als tekstverwerkers en spelletjes brengen vaak gratis zogenoemde bètaversies in omloop en recruteren daarmee een legertje gratis testers. Aan de hand van hun bevindingen komt de definitieve versie tot stand. Die ligt in de winkel met de nadrukkelijke boodschap aan de consument dat hij niet moet zeuren als er nog fouten in zitten.

Tussen de twee extremen zit een scala aan kleine en grote systemen die voor specifieke doeleinden zijn gebouwd, van het besturingsprogramma van de magnetron tot de enorme databases van de bank. De magnetron is nog vrij simpel te beproeven, maar het testen van grotere programma's kost zoveel tijd en geld, dat de software vaak te snel in het diepe wordt gegooid. Dit is de bron van de eindeloze reeks berichten over 'moeilijkheden bij de invoer van een nieuw computersysteem'. In bijna alle gevallen betekent dat: te haastig geweest en over de eigen voeten gestruikeld. Tegen programmeerfouten helpt alleen wat in het algemeen helpt tegen fouten: opletten en geconcentreerd blijven. Er wordt weliswaar gewerkt aan methoden om de juistheid van programma's te 'bewijzen', maar die kijken alleen of het uiteindelijke programma overeenkomt met de specificaties. Een fout als die in de Therac 25 zou zo waarschijnlijk aan het licht komen. De meeste fouten worden echter in het voortraject gemaakt: tijdens het exact specificeren van wat het programma moet doen. Dat blijft het werk van mensen, die het opnemen tegen een wirwar van mogelijkheden waarin het bos regelmatig achter de bomen schuil gaat. Ieder programma bevat tientallen, zoniet honderden fouten, maar zelfs het best getrainde oog kijkt over de meeste fouten heen. Sterker nog, er zijn fouten die steeds worden herhaald, terwijl duizenden mensen toekijken en niemand iets

door heeft. De Amerikaanse overheid vond onlangs een fout uit 1972, die betekende dat sinds die tijd bijna anderhalf miljard te weinig aan pensioenen was uitgekeerd. De vierhonderdduizend gedupeerden krijgen hun geld alsnog, maar voor 57.500 van hen komt dit te laat. Zij zijn intussen overleden.

Christian Jongeneel

MICROSOFT EXECUTIVE E-MAIL, Oct. 2, 2002

Connecting with Customers

I spend a lot of my time thinking about how Microsoft can do a better job of serving its customers. I'm convinced that we need to do more to establish and maintain broad connections with the millions of people who use our products and services around the world. We need to more thoroughly understand their needs, how they use technology, what they like about it, and what they don't. I'd like to share with you some of what we've recently begun to do and are planning for in the future to better connect with our customers.

Software and Snack Food
In my career, I've worked at only one other place besides Microsoft. I marketed brownie mix and blueberry muffin mix for one of the largest consumer products companies. I'm glad I decided to join Microsoft 22 years ago, when it was a little software startup, but I have great admiration for successful consumer businesses, and I believe Microsoft can learn from them. Behind the leading brands are companies that really know their customers. These firms devote a great deal of time and energy to gaining an intimate understanding of consumers, their reactions to every aspect of products, and how those products fit into their lives. Even so, not every new grocery or drug-store item succeeds. But by using the huge volume of data that feeds back from the daily purchase decisions of millions of consumers, marketers manage over time to figure out what consumers want in cake mix, soft drinks, shampoo, and so on. And these same products often go on satisfying consumers for decades.

Satisfying customers is what it's all about with technology products, too. And customers expect the same high quality and reliability in computing devices and software as they do in consumer products. But meeting their expectations is much harder, and not just because information technology is more complex and interdependent. The challenge has more to do with the flexibility of technology and its continual, rapid advance. To take advantage of this and expand what people can do with hardware and software, computer products must constantly evolve. As a result, products are seldom around long enough in one form to be fully time-tested, let alone perfected. And customers continually come up with new uses for their technology, new combinations and configurations that further complicate technology companies' efforts to ensure a satisfying experience, free of hiccups and glitches.

If technology products are to approach the satisfying consistency of consumer staples — and clearly they should — then we in the industry need a more detailed knowledge of customers' experiences with our products. We must do a better job of connecting with customers. For a company such as Microsoft, with many millions of customers around the world, the connections must be very broad. While we are working to deepen our relationships with enterprise and other business customers, we also need to make innumerable, daily connections with the very wide array of people who use our products — consumers, information workers, software developers and information technology professionals.

In the past year, we specifically identified some near-term objectives on the road to further product improvements and greater customer satisfaction. Among them:
- *Obtain much more feedback from our customers about their experience;*
- *Offer customers easier, more consistent ways to update their products;*
- *Provide customers with more effective, readily available support and service.*

We have a long way to go, but we're excited about the results so far from some of our recent efforts. I'd like to share just one great example, and then I'll tell you how you can learn more about what we're doing along these lines.

A New Pipeline for Customer Feedback

Let's acknowledge a sad truth about software: any code of significant scope and power will have bugs in it. Even a relatively simple software product today has millions of lines of code that provide many places for bugs to hide. That's why our customers still encounter bugs despite the rigorous and extensive stress testing and beta testing we do. With Windows 2000 and Windows XP, we dramatically improved the stability and reliability of our platform, and we eliminated many flaws, but we did not find all the bugs in these or other products. Nor did we find all the software conflicts that can cause applications to freeze up or otherwise fail to perform as expected.

The process of finding and fixing software problems has been hindered by a lack of reliable data on the precise nature of the problems customers encounter in the real world. Freeze-ups and crashes can be incredibly irritating, but rarely do customers contact technical support about them; instead, they close the program. Even when customers do call support and we resolve a problem, we often do not glean enough detail to trace its cause or prevent it from recurring.

To give us better feedback, a small team in our Office group built a system that helps us gather real-world data about the causes of customers' problems — in particular, about crashes. This system is now built into Office, Windows, and most of our other major products, including our forthcoming Windows .NET Servers. It enables customers to send us an error report, if they choose, whenever anything goes wrong.

There are risks in offering this option to have software "phone home" like E.T. One risk is that error reporting could compound a customer's irritation over the error itself. We therefore worked hard to make reporting simple and quick. We developed a special format, called a "minidump," to minimize the size of the report so that it can be transferred in a few seconds with a single mouse click.

Also, customers may wonder what we do with their reports and whether their privacy is protected. We use advanced security technologies to help protect these error reports, which are gathered on a cluster of dedicated Microsoft servers and are used for no other purpose than to find and fix bugs. Engineers look at stack details, some system information, a list of loaded modules, the type of exception, and global and local variables.

We've been amazed by the patterns revealed in the error reports that customers are sending us. The reports identify bugs not only in our own software, but in Windows-based applications from independent hardware and software vendors as well. One really exciting thing we learned is how, among all the software bugs involved in reports, a relatively small proportion causes most of the errors. About 20 percent of the bugs cause 80 percent of all errors, and — this is stunning to me — one percent of bugs cause half of all errors.

With this immensely valuable feedback from our customers, we're now able to prioritize debugging work on our products to achieve the biggest improvement in customers' experience. And as the work proceeds based on this new source of systematic data, the improvement will be dramatic. Already, in Windows XP Service Pack 1, error reporting enabled us to address 29 percent of errors involving the operating system and applications running on it, including a large number of third-party applications. Error reporting helped us to eliminate more than half of all Office XP errors with Office XP Service Pack 2.

Work continues to find and fix remaining bugs in these and other existing products, but error reporting is now also helping us to resolve more problems before new products are released. Visual Studio .NET, released last February, was one of our first products to benefit from the use of error-reporting data throughout its beta testing. Error reporting enabled us to log and fix 74 percent of all crashes reported in the first beta version. Many other problems were caught and eliminated in subsequent testing rounds.

And we're not keeping this great tool to ourselves. We're working with independent hardware and software vendors to help them use our error-reporting data to improve their products, too. Some 450 companies have accessed our database of error reports related to their drivers, utilities and applications. Marked decreases in some types of errors have followed. Those involving third-party firewall software, for example, have dropped 67 percent since the first of the year. Also, we've created software that enables corporations to redirect error reports to their own servers, so that administrators can find and resolve the problems that are having the most impact on their systems.

This Is Just the Beginning

We're working to make error reporting a much more supple tool that provides helpful information to customers while enabling us to improve their experience in new ways. As we understand more errors, we're adding an option for customers to go to a Web site where they can learn more about and even fix the errors they report. In the future we want to enable customers to look up the history of their error reports and our efforts to resolve them. And we're trying to create easy ways for customers to send us more nuanced feedback about their experience with our products — not only about crashes, but also about features that don't work the way or as easily as people would like.

Microsoft Error Reporting is just one of the ways in which we're trying to create broader customer connections. Another is through our software update and management services, which make it easy for customers to keep their software current. We're also making significant changes in our product service and support to enhance their value, and to speed the resolution of customer problems. Soon we will commit to a new policy that will give customers greater clarity and confidence about our support for products through their lifecycles.

There's much more I would like to share with you about these and other initiatives on behalf of customers, but I wanted to be (relatively) brief. If you would like to know more, here are information and links to help you drill down even further.

Ultimately, we're trying to change how software developers do their jobs on a daily basis. We're working to establish more of a direct, interactive connection between developers and customers, leading to better software and happier customers. To get there, we intend to listen even more closely to our customers, consult with them regularly, and be more responsive. This is the message I am sending to all of Microsoft's employees, and it is my commitment to you.

Thanks for taking the time to read this.

Steve Ballmer

Bijlage 2 Redundantie door voting

De bewaking van een proces gaat met sensoren, bijvoorbeeld temperatuur, druk, niveau sensoren. Indien een kritische waarde wordt overschreden moet een veiligheidsactie worden uitgevoerd. Behalve dat in het proces wat fout gaat, kan er ook wel eens iets mis zijn met de sensoren zelf en de verwerking van de signalen. Hierbij zijn twee mogelijkheden de veiligheidsactie wordt **niet** uitgevoerd of de veiligheidsactie wordt onnodig uitgevoerd door zoals men dat noemt een **vals** signaal.

Ter voorkoming van geen actie of een vals signaal bouwt men bij de beveiliging redundantie (reserve) in. Dit bereikt men door zogenaamde *voting* configuraties. Voting (eigenlijk volgens de engelse term kiezen) wil zeggen een besluit nemen op basis van signalen. Dit kan een enkelvoudig signaal zijn of een meervoudig signaal. Het gebruik maken van meervoudige signalen verhoogt de veiligheid en/of de betrouwbaarheid van een veiligheidssysteem.

De uitdrukking voor een zekere voting configuratie is M-out-of-N of kortweg MooN. Het betekent meer formeel gezegd: M van de N elementen / componenten in de configuratie moeten functioneren om er voor te zorgen dat het subsysteem kan functioneren. M geeft het aantal signalen dat nodig is voor een actie en N wil zeggen het aantal sensoren die het signaal afgeven.

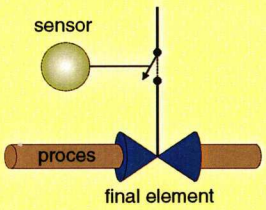
M-out-of-N of MooN

- *Een votingconfiguratie **1oo1** betekent dat M=1 (één signaal is voldoende voor het sluiten) en N=1 (één sensor).*
- *Votingconfiguratie **1oo2** betekent M=1 (één signaal is voldoende voor het sluiten) en N=2 (twee sensoren).*
- *Votingconfiguratie **2oo2** betekent M=2 (signalen) en N=2 (twee sensoren)*

Votingconfiguraties komen voor in het gehele *safety instrumented system* zowel binnen een groep van componenten / onderdelen, maar ook (soms) tussen dergelijke groepen onderling. Een aantal veel voorkomende *voting* configuraties die men tegenkomt in de (petro)chemische procesindustrie worden hieronder getoond en kort toegelicht.

Op de volgende pagina's worden de verschillende votingconfiguraties geïllustreerd aan de hand van vereenvoudigde schema's. We gaan uit van 1, 2 of 3 sensoren die bij een kritisch waarde van bijvoorbeeld te hoge temperatuur het signaal afgeeft dat een afsluiter (final) element moeten sluiten. Voor de gedachtevorming stellen we dat de afsluiter normaal wordt opengehouden door luchtdruk. Als het signaal komt sluiten dan valt de druk weg (verbreking toevoer) en sluit de afsluiter door veerdruk.

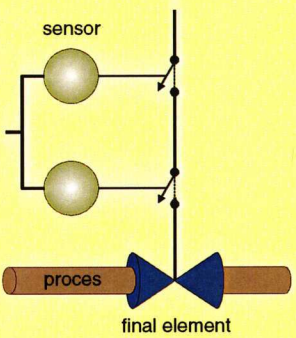
1oo1 (één signaal van één sensor)



Wanneer de sensor een 'hazard' signaal krijgt van het proces, zal deze de schakelaar openen en zal er geen druk meer staan op de afsluiter (wegnemen van energie) en sluit de afsluiter of het *final element*, waardoor de configuratie de beoogde veiligheidsfunctie heeft uitgevoerd.

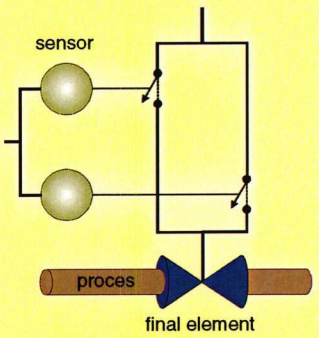
Merk op: dat bij deze configuratie geen sprake is van redundantie. Er is slechts één signaal van één sensor.

1oo2: (één signaal van twee sensoren)



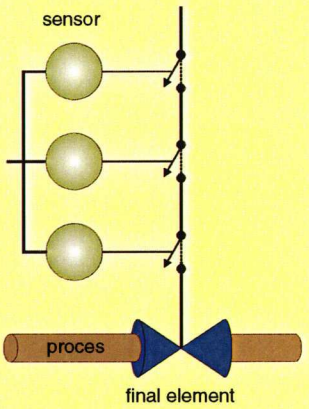
Sluiten treedt op als één van de beide sensoren het sluitsignaal geeft. Hier is vooral een veiligheidsgedachte voor de configuratie. Sluiten staat voorop.

2oo2: (twee signalen van twee sensoren



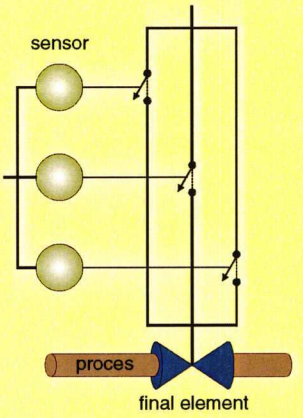
Het verschil met de configuratie hiernaast is dat beide sensoren een sluitsignaal moeten geven. De configuratie heeft een veiligheidsfunctie maar zal minder gauw op een vals signaal sluiten, dus ook betrouwbaarheid wordt meegewogen.

1oo3: (één signaal van drie sensoren)



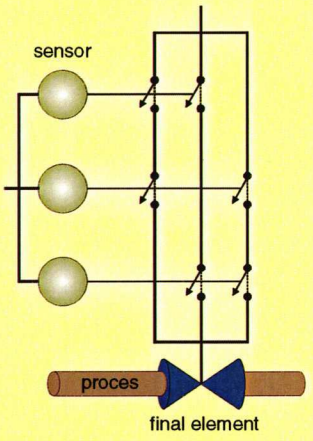
Bij deze configuratie is het voldoende als één van de drie sensoren een signaal afgeeft voor sluiten.

3oo3: (drie signalen van drie sensoren)



De tegenhanger van de configuratie hiernaast. Nu moeten alle drie de sensoren het signaal sluiten geven.

2oo3: (twee signalen uit drie sensoren)



Het is hier voldoende of anders gezegd tenminste twee sensoren moeten een sluitsignaal geven.

De configuratie hiernaast (2oo3) komt het meeste tegemoet aan zowel de eis van veiligheid als betrouwbaarheid.

Bijlage 3 Analysetechnieken

Om een uitspraak te kunnen doen over het Safety Integrity Level van (een *safety function* uitgevoerd door) een *safety instrumented system*, zijn diverse kwalitatieve en kwantitatieve technieken beschikbaar voor hardware en kwalitatieve technieken voor software. Deze technieken variëren in complexiteit (van redelijk eenvoudig tot behoorlijk complex) en toepasbaarheid.

In diverse internationale standaarden worden de kwalitatieve analysetechnieken expert analyse en Failure Mode and Effects analysis (FMEA) aanbevolen. Van deze standaarden kan ook worden afgeleid dat veelgebruikte kwantitatieve analysetechnieken zijn: Simplified Equations, Reliability Block Diagrams, Fault Tree analysis en Markov analyse. Deze technieken worden hieronder kort besproken. Tevens wordt ingegaan op faaldata voor een kwantitatieve inschatting van de Safety Integrity van hardware.

B3.1 Kwalitatieve analysetechnieken

De kwalitatieve analysetechnieken worden meestal gebruikt als input voor de kwantitatieve analysetechnieken.

Expert mening of expert analysis

Expert analysis[1] maakt gebruikt van ervaringen van experts die ze hebben opgedaan tijdens de ontwikkeling van eerdere *safety instrumented systems*. Deze analysetechniek kan worden ondersteund door bijvoorbeeld checklists, ontwerprichtlijnen en praktijkvoorschriften (*codes of practice*). Het resultaat is meestal een rangorde van de *safety integrity* van de geanalyseerde systemen.

Failure Mode and Effects analysis

Failure Mode and Effects analysis oftewel 'FMEA'[2] maakt gebruik van een zogenaamde bottom-up benadering door voor ieder component / onderdeel in het SIS het volgende te specificeren:

- Faalmodes inclusief de faaloorzaken
- Het gevolg van het falen op het bovenliggende functioneel niveau (bijv. voor het falen van een component: het gevolg op *subsystem* niveau)
- Het gevolg van het falen op systeemniveau
- Een kwalitatieve aanduiding van de kans van falen en de effecten ervan
- Eventueel: opmerkingen en te ondernemen acties

1. Zie ook literatuurreferenties [CCP93, DIN90, DIN92, DIN94A, DIN94B].
2. [MIL84, IEC88].

Een voorbeeld van een mogelijke FMEA tabel wordt getoond in tabel B3.1

Tabel B3.1 Voorbeeld gedeeltelijkingevulde FMEA tabel.

Failure Mode and Effect Analysis							
Component Informatie			Informatie over falen				
1	2	3	4	5	6	7	8
Naam	Code	Functie	Faalmode	Oorzaak	Gevolg op systeem-niveau	Faal frequentie	Opmerkingen + te onderne-men actie
Druk-sensor	DS1	Meten druk	Kan geen druk meten	Corrosie	Sensor kan proces niet stilleggen wanneer nodig		
				Kort-sluiting	Sensor kan proces niet stilleggen wanneer nodig		
			Geeft 'hazard' signaal ónder ingestelde waarde	Verkeerde waarde ingesteld	Sensor legt proces stil terwijl niet nodig		
				Trillingen	Sensor legt proces stil terwijl niet nodig		

Afgeleid van de FMEA zijn bijvoorbeeld de analysemethoden: Failure Mode Effects and Criticality analysis (inschatting van de ernst en waarschijnlijkheid op basis van een ranking tabel) en Failure Modes, Effects and Diagnostics analysis. .

B3.2 Kwantitatieve analysetechnieken

Berekeningsmethoden

De kwantitatieve analysetechnieken kunnen worden gebruikt voor de berekening van de kans dat een safety instrumented system zijn functie niet uitvoert. Deze kans wordt ook wel de *probability of failure* on demand (PFD) genoemd.

Uit de internationale standaarden kan worden afgeleid dat veelgebruikte kwantitatieve analysetechnieken voor hardware *safety integrity* zijn[1]:

- Simplified equations
- Reliability Block Diagrams
- Fault Tree analysis
- Markov analysis

Er zijn diverse (software) tools (commercieel) beschikbaar die het modelleren, rekenen en/of analyseren met behulp van deze kwantitatieve technieken vergemakkelijken. Met name voor de complexere technieken als Fault Tree analysis en Markov analysis zijn vele tools beschikbaar. Met betrekking tot Fault Tree analysis zijn de meeste tools zogenaamde rekentools: het model wordt ingevoerd door gebruiker, en het rekenwerk (en enige controles) wordt uitgevoerd door een software programma .

1. Nadere informatie in: [AAR89, IEC91, CCP93].

Met betrekking tot Markov analysis kunnen de tools onderverdeeld worden in:

- Databases: het tool bevat voorgeprogrammeerde modellen van een aantal (delen van) configuraties. Met gebruikmaking van data ingevoerd door de gebruiker kunnen deze modellen doorgerekend worden.
- Modelleer tools: op basis van een (vaak vereenvoudigd) model van de configuratie wordt een Markov model opgesteld en, met gebruikmaking van data ingevoerd door de gebruiker, doorgerekend.
- Reken tools: het Markov model wordt volledig opgesteld en ingevoerd door de gebruiker en, met gebruikmaking van data ingevoerd door de gebruiker, doorgerekend

Over de toepasbaarheid van de diverse tools (voor de diverse analysetechnieken) wordt discussie gevoerd. De vier genoemde analysetechnieken worden kort toegelicht.

Simplified Equations

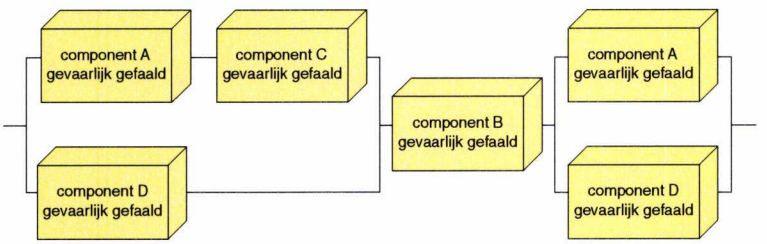
In het technical report ISA-dTR84.0.02 (part 2) behorende bij de ISA-S84.01 standaard staan zogenaamde *Simplified Equations* (vereenvoudigde wiskundige vergelijkingen), die een indicatie geven over de gemiddelde PFD (probability of failure on demand, faalkans op aanspraak) van een bepaalde voting configuratie. Deze vergelijkingen worden gegeven voor de *voting* configuraties 1oo1, 1oo2, 1oo3, 2oo2, 2oo3 en 2oo4 (zie voor een toelichting op *voting* configuraties bijlage 2).

Reliability Block Diagrams

Een Reliability Block Diagram[1] is een schematische weergave van de functionele afhankelijkheid van de componenten van een *safety instrumented system*. Dat wil zeggen dat het diagram niet de fysische stromen en verbindingen laat zien, maar de logische stromen en verbindingen. Het diagram geeft de verschillende mogelijke paden aan waarlangs deze stromen kunnen lopen; wanneer er in het diagram van links naar rechts een doorgang gevonden wordt (een pad), werkt het systeem.

Een component kan meerdere malen voorkomen in het diagram, afhankelijk van zijn functie. Componenten kunnen parallel of in serie staan, of een combinatie hiervan. Hierbij wordt er vanuit gegaan, dat de componenten niet te repareren zijn en dat de fouten onafhankelijk zijn.

Ter illustratie wordt onderstaand een voorbeeld van een Reliability Block Diagram getoond.



Figuur B3.1 Voorbeeld van een Reliability Block Diagram, ter illustratie.

1. Nadere informatie in: [AAR89, IEC91, CCP93].

Per faalmode van het gehele *safety instrumented system*, dient een apart Reliability Block Diagram te worden opgesteld. Een voorbeeld van een dergelijke faalmode is het niet meer kunnen functioneren wanneer het beveiligde proces daarom vraagt (vanwege een gevaarlijke procesconditie).

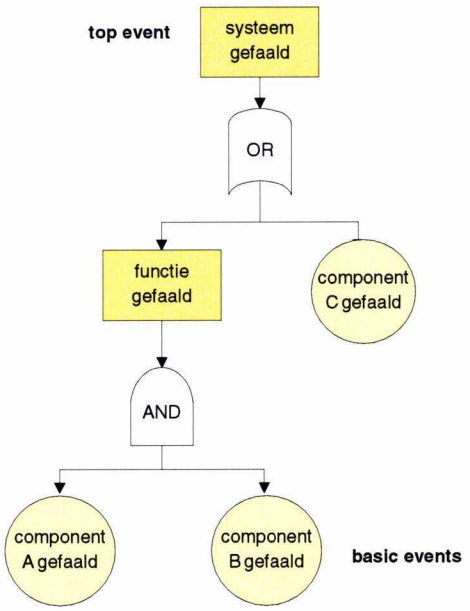
Een Reliability Block Diagram kan geanalyseerd worden met behulp van standaard kansrekening. Dit rekenwerk kan vereenvoudigd worden door gebruikmaking van zogenaamde Minimal Cut Sets. Dit zijn combinaties van componenten zodanig dat, wanneer al deze componenten falen, het gehele systeem faalt en wanneer één van deze componenten functioneert, het gehele systeem functioneert.

Fault Tree Analysis

Het doel van *Fault Tree Analysis*[1] is het vinden van die combinaties van oorzaken die leiden tot een zekere (ongewenste) gebeurtenis, oftewel de *top event*, door het opstellen van een zogenaamde foutenboom. Deze ongewenste gebeurtenis kan bijv. een ongeluk zijn, of een probleem. Bijvoorbeeld het niet sluiten van kleppen, wanneer de druk te hoog wordt. Ieder *top event* vereist een eigen foutenboom.

Fault tree analysis is, in tegenstelling tot FMEA, een "top down" methode. Het begint met een top event, en analyseert deze om te bekijken welke gebeurtenissen leiden tot dit *top event*. Elke van deze gebeurtenissen kunnen op hun beurt uitgewerkt worden tot op het niveau van zogenaamde basis oorzaken (*basic events*).

Een foutenboom wordt vaak gebruikt om grafisch weer te geven welke gebeurtenissen en oorzaken leiden tot een zeker *top event* (ongewenste gebeurtenis). Foutenbomen kunnen echter niet omgaan met reparatie of testen van (onderdelen van) *safety instrumented systems*. Ter illustratie is een voorbeeld van een foutenboom weergegeven in 3.2.



Figuur B3.2 Voorbeeld van een foutenboom.

1. Meer informatie is te vinden in de literatuureferenties [IEC90, AND93, KUM96].

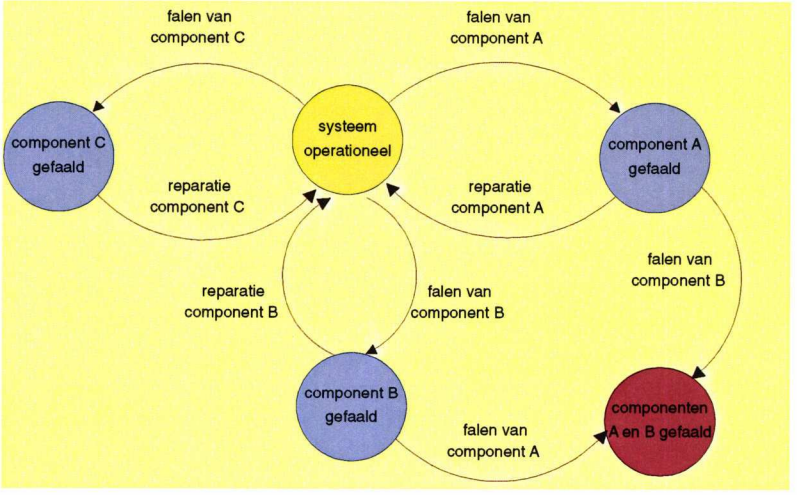
Er worden in een foutenboom diverse symbolen gebruikt om diverse gebeurtenissen of relaties hiertussen aan te geven. Wanneer een gebeurtenis bijv. veroorzaakt wordt door twee of meer oorzaken kan door middel van zogenaamde AND- en OR-gates aangegeven worden hoe deze oorzaken zich tot elkaar verhouden.

Fault Tree Analysis is een kwalitatieve analyse die echter ook kwantitatieve mogelijkheden biedt. Per basisoorzaak (*basic event*) kan een faalkans worden aangegeven. Met behulp van kansrekening kan dan een schatting gemaakt worden van de kans op het *top event*.

Markov analysis

Markov analysis[1] begint met het ontwikkelen van een zogenaamd Markov model. Dit model geeft grafisch alle mogelijke toestanden van het systeem weer, met behulp van cirkels en pijlen. De cirkels stellen de specifieke toestanden voor waarin het systeem zich kan bevinden. De overgangen van het systeem tussen deze toestanden worden weergegeven met behulp van pijlen.

Een systeem kan zich op een zeker tijdstip in slechts één toestand (één cirkel) bevinden. Een overgang van de ene naar de andere toestand stelt falen, reparatie of testen van een of meerdere componenten in het systeem voor. Iedere overgang wordt gekarakteriseerd door een overgangskans (transition probability) of een overgangsfrequentie (*transition rate*). Ter illustratie wordt een voorbeeld van een Markov model weergegeven in figuur B3.3.



Figuur B3.3 Voorbeeld van een Markov model.

Met behulp van de kansen gekoppeld aan de overgangen (transities) in het Markov model, kan de kans worden berekend (als functie van de tijd) dat het systeem zich in een zekere toestand bevindt. Om de probability of *failure on demand* (PFD) te berekenen (en zo het *safety integrity level* te kunnen bepalen), dienen de kansen opgeteld te worden van die toestanden waarin het systeem (gedetecteerd of ongedetecteerd) zijn functie niet meer kan uitvoeren wanneer nodig.

---

1. Zie ook de literatuurreferenties[GOB92, BOY95, GOB98B, ROU01].

B3.3 Faaldata

Teneinde een kwantitatieve inschatting te kunnen maken van de *safety integrity* met betrekking tot de hardware, dienen faaldata beschikbaar te zijn. Onder faaldata wordt verstaan: getallen en informatie die het falen van hardware (onderdelen) van het SIS karakteriseren. Voorbeelden hiervan zijn: faalfrequenties (*failure rates*) voor de diverse faalmodes per type component, reparatieduur, mate van dekking van periodieke test. Voor dit laatste wordt ook wel de engelse term *proof test coverage factor* gebruikt.

In het ideale geval voldoen deze data aan de volgende eisen:

- De data dienen relevant en valide te zijn. Dat wil zeggen de data is van toepassing op praktisch identieke componenten, faaloorzaken en gebruiksomgeving van de desbetreffende fout.
- De data dienen betrouwbaar te zijn. Dat wil zeggen gebaseerd op voldoende waarnemingen van de soortgelijke fout.
- De data dienen verifieerbaar en traceerbaar te zijn. Dat wil zeggen de achtergrond en bron van de data dienen bekend te zijn en (bewijsbaar) valide (juist verzameld en bewerkt).

Faaldata wordt meestal verkregen door een 'vertaling' van praktijkdata in bruikbare data.

Veelgebruikte bronnen voor data zijn handboeken en databases. Hierin is meestal een verzameling van data weergegeven, per component en/of subsysteem, welke een indicatie geeft met betrekking tot de te gebruiken getallen zoals faalfrequenties en reparatietijden. Databases bevatten meestal data die in praktijksituaties zijn verzameld of resultaten van (praktijk)testen. De handboeken zijn vaak geschreven voor specifieke industrieën zoals telecommunicatie en off-shore[1].

Figuur B3.4 laat zien hoe faaldata kan worden gepresenteerd in een handboek. Het voert te ver om de gehele datasheet te bespreken, echter wel een opmerking over de cijfers in de tabel. De cijfers zijn gebaseerd op een beperkte populatie (de verzameling systemen of installaties waar men gegevens van heeft verzameld). In dit geval bestaat de populatie slechts uit 5 compressoren in 2 installaties. Het niet willen starten (failed to start) is éénmaal voorgekomen. Met dat getal wordt dan een faalfrequentie berekend en dit kan, omdat de populatie klein is niet anders zijn dan een schatting met een lage nauwkeurigheid. Helaas is dat het geval bij veel faaldata.

1. Voorbeelden van veelgebruikte handboeken zijn MIL-HDBK-217F [MIL92], het Offshore Reliability Data Handbook OREDA [ORE97], en het handbook of Reliability Data for Components used in Telecommunications Systems [BT87].

Taxonomy no 1.1.1.1.1		Item Machinery Compressors Centrifugal Electric Motor Driven (100-1000) kW								
Population	**Installations**	**Aggregated time in service (10^6 hours)**						**No of demands**		
5	2	**Calendar time *** 0.1248		**Operational time** 0.0832						
Failure mode		**No of**	**Failure rate (per 10^6 hours)**					**Active**	**Repair (manhours)**	
		fail.	**Lower**	**Mean**	**Upper**	**SD**	**MLE**	**rep.hrs**	**Min**	**Mean**	**Max**
Critical		**23*** **23**	**1.31** **2.02**	**217.71** **471.18**	**827.93** **1806.90**	**304.49** **665.33**	**184.33** **276.36**	**10.0**	**0.5**	**24.3**	**186.3**
Failed to start		1* 1	0.94 0.29	8.42 17.10	22.20 61.58	7.02 22.41	8.01 12.02	-	13.0	13.0	13.0
Fail while running		14* 14	0.97 1.28	132.09 285.42	499.13 1093.54	183.39 402.61	112.20 168.22	10.0	0.5	24.0	186.3
Unknown		1* 1	0.94 0.29	8.42 17.10	22.20 61.58	7.02 22.41	8.01 12.02	-	11.4	11.4	11.4
Vibration		7* 7	0.71 0.70	65.50 140.94	243.34 538.64	89.14 198.24	56.10 84.11	-	0.5	28.5	117.5

Figuur B3.4 Voorbeeld van (deel van) tabel in Oreda handboek [ORE97].

Om onzekerheid in faaldata te verkleinen en er zeker van te zijn dat faaldata geschikt zijn heeft geleid tot het *proven in use* concept (bewijs uit ervaring) zoals verwoord in de standaard IEC 61508. Doel van het concept is om te kunnen bewijzen dat de gebruikte data inderdaad betrouwbaar en valide zijn[1]. In het "proven in use" concept worden kwantitatieve eisen gesteld aan de hoeveelheid data en de kwaliteit ervan alvorens deze gebruikt mag worden in analyses. Bijvoorbeeld: de data moeten afkomstig zijn van minimaal 10 verschillende installaties.

1. Zie ook [IEC00, IEC01].

Bijlage 4 Verkorte checklists

Opgemerkt wordt dat door anderen reeds vele checklists zijn vervaardigd en worden toegepast. Bij nadere bestudering daarvan blijkt dat deze checklists niet erg toegankelijk zijn en veel achtergrondkennis vergen. Het is vooral om deze reden dat de door TNO opgestelde checklists zich beperken tot hoofdpunten die met inzichten opgedaan met dit boekje kunnen worden beoordeeld.

Overzicht beoordeling verkorte checklist 1

Overzicht van de checklist:

Hoofdpunten	Beoordelingen		
	Voldoende	Onvoldoende	Opmerkingen
A: Veiligheidsrapport	☐	☐	
B: Functionele Specificatie	☐	☐	
C: Gevaren- en risicoanalyse	☐	☐	
D: Safety plan	☐	☐	
E: Kwaliteitsplan	☐	☐	
F: Configuratie management procedures	☐	☐	
G: Functional Safety Assessment	☐	☐	

A: Veiligheidsrapport

Te checken punten:	**Antwoorden**
1) Gevaren van het proces omschreven?	Ja ☐ Nee ☐ Opmerking:
2) Bijgewerkt?	Ja ☐ Nee ☐ Opmerking:
3) SIL vastgesteld?	Ja ☐ Nee ☐ Opmerking:
Beoordeling: Dit onderdeel is als *onvoldoende* aan te merken wanneer één van bovenstaande punten met *nee* wordt beantwoord.	☐ Voldoende ☐ Onvoldoende Opmerking:

B: Functionele Specificatie van Proces en Installatie

Te checken punten:	Antwoorden
1) Op onderdeel niveau uitgewerkt?	Ja ☐ Nee ☐ Opmerking:
2) In detail uitgewerkt met onder ander schema's en tabellen?	Ja ☐ Nee ☐ Opmerking:
3) P&ID in detail opgesteld?	Ja ☐ Nee ☐ Opmerking:
4) P&ID bijgewerkt en in overstemming met functionele specificatie?	Ja ☐ Nee ☐ Opmerking:
Beoordeling: Dit onderdeel is als *onvoldoende* aan te merken wanneer één van bovenstaande punten met *nee* wordt beantwoord.	☐ Voldoende ☐ Onvoldoende Opmerking:

C: Gevaren- en risicoanalyse	
Te checken punten:	**Antwoorden**
1) Uitgevoerd door deskundige team?	Ja ☐ Nee ☐ Opmerking:
2) Aanbevelingen geïmplementeerd?	Ja ☐ Nee ☐ Opmerking:
Beoordeling: Dit onderdeel is als *onvoldoende* aan te merken wanneer één van bovenstaande punten met *nee* wordt beantwoord.	☐ Voldoende ☐ Onvoldoende Opmerking:

D: Safety plan	
Te checken punten:	Antwoorden
1) Opgesteld aan het begin van het ontwerpproces en bijgehouden?	Ja ☐ Nee ☐ Opmerking:
2) Documentatie en management procedures beschreven?	Ja ☐ Nee ☐ Opmerking:
3) Onafhankelijk getoetst?	Ja ☐ Nee ☐ Opmerking:
4) Audits uitgevoerd op correcte uitvoering?	Ja ☐ Nee ☐ Opmerking:
Beoordeling: Dit onderdeel is als *onvoldoende* aan te merken wanneer één van bovenstaande punten met *nee* wordt beantwoord.	☐ Voldoende ☐ Onvoldoende Opmerking:

E: Kwaliteitsplan

Achtergrond informatie:

Te checken punten:	**Antwoorden**
1) In overstemming met het algemene kwaliteitsplan?	Ja ☐ Nee ☐ Opmerking:
2) Audits uitgevoerd op inhoud?	Ja ☐ Nee ☐ Opmerking:
3) Auditor heeft kwaliteitsplan geaccepteerd?	Ja ☐ Nee ☐ Opmerking:

Beoordeling: Dit onderdeel is als *onvoldoende* aan te merken als één van bovenstaande punten met *nee* wordt beantwoord.	☐ Voldoende ☐ Onvoldoende Opmerking:

F: Configuratie management procedures	
Te checken punten: 1) Uitgewerkt?	**Antwoorden** Ja ☐ Nee ☐ Opmerking:
Beoordeling: Dit onderdeel is als *onvoldoende* aan te merken wanneer het bovenstaande punt met *nee* wordt beantwoord.	☐ Voldoende ☐ Onvoldoende Opmerking:

G: Functional Safety Assessment	
Te checken punten:	**Antwoorden**
1) Dekkend voor alle fasen van de safety life cycle?	Ja ☐ Nee ☐ Opmerking:
2) Vastgesteld SIL (L of H)?	Ja ☐ Nee ☐ Opmerking:
Beoordeling: Dit onderdeel is als *onvoldoende* aan te merken als één van bovenstaande punten met *nee* wordt beantwoord.	☐ Voldoende ☐ Onvoldoende Opmerking:

Overzicht beoordeling verkorte checklist 2

Overzicht van de checklist:

Hoofdpunten	Beoordelingen		
	Voldoende	Onvoldoende	Opmerkingen
A: Onafhankelijkheid	☐	☐	
B: Juist niveau vakbekwaamheid	☐	☐	
C: Functional Safety Assessment	☐	☐	
D: Overall bewijs	☐	☐	

A: Onafhankelijkheid	
Te checken punten: 1) In overstemming met Safety Integrity Level (SIL)? (zie tabel achterin de bijlage)	**Antwoorden** Ja ☐ Nee ☐ Opmerking:
Beoordeling: Dit onderdeel is als *onvoldoende* aan te merken wanneer het bovenstaande punt met *nee* wordt beantwoord.	☐ Voldoende ☐ Onvoldoende Opmerking:

B: Juist niveau vakbekwaamheid

Te checken punten:	Antwoorden
1) Toepassingsgebied?	Ja ☐ Nee ☐ Opmerking:
2) Technologie?	Ja ☐ Nee ☐ Opmerking:
3) Safety engineering?	Ja ☐ Nee ☐ Opmerking:
4) Veiligheidsvoorschriften?	Ja ☐ Nee ☐ Opmerking:
5) Ervaring (certificaat)?	Ja ☐ Nee ☐ Opmerking:

Beoordeling:	
Dit onderdeel is als *onvoldoende* aan te merken wanneer één van bovenstaande punten met *nee* wordt beantwoord.	☐ Voldoende ☐ Onvoldoende Opmerking:

C: Functional Safety Assessment

Te checken punten:	**Antwoorden**
1) Rapporten mede ondertekend door verantwoordelijk manager?	Ja ☐ Nee ☐ Opmerking:
2) Geconstateerde tekortkomingen door SIS-ontwerper opgelost?	Ja ☐ Nee ☐ Opmerking:
Beoordeling: Dit onderdeel is als *onvoldoende* aan te merken wanneer één van bovenstaande punten met *nee* wordt beantwoord.	☐ Voldoende ☐ Onvoldoende Opmerking:

D: Overall bewijs	
Te checken punten: 1) Wordt het overall bewijs van veiligheid bekrachtigd door de onafhankelijke beoordelaar?	Antwoorden Ja ☐ Nee ☐ Opmerking:
Beoordeling: Dit onderdeel is als *onvoldoende* aan te merken wanneer het bovenstaande punt met *nee* wordt beantwoord.	☐ Voldoende ☐ Onvoldoende Opmerking:

Bijlage A: Onafhankelijkheidstabel

Tabel 1: Onafhankelijk beoordelaar, afhankelijk van SIL

Minimum level of independence	Safety Integrity Level			
	1	2	3	4
Independence person	HR	HR[1]	NR	NR
Independence department	--	HR[2]	HR[1]	NR
Independence organization (see note 2 of 8.2.12)	--	--	HR[2]	HR

Note – See 8.2.12 (including notes), 8.2.13 and 8.2.14 for details on interpreting this table.

- *NR = Not Recommended: niet aanbevolen (afgeraden)*
- *HR = Highly Recommended: sterk aanbevolen*
- *HR1 of HR2: voor een onderscheid in hiervoor genoemde zaken wordt de aanbeveling HR (highly recommended) verder ingedeeld in HR1 en HR2. Er wordt gebruik gemaakt van aanbeveling HR2 wanneer er sprake is van:*
  - *Complexheid en onbekendheid van het ontwerp*
  - *Onbekendheid van de toe te passen technologie en hierdoor gebrek aan ervaring met het ontwerp*
  - *Minder standaardisatie in het ontwerp*