Rijkswaterstaat
Ministerie van Verkeer en Waterstaat

Handleiding TOPAAS

Een structurele aanpak voor faalkansanalyse van software intensieve systemen

Datum 10 januari 2013
Status Definitief

Handleiding TOPAAS

Een structurele aanpak voor faalkansanalyse van software intensieve systemen

Datum 10 januari 2013
Status Definitief

Colofon

Uitgegeven door	RWS Dienst Infrastructuur
Informatie	S.E. van Manen
Telefoon	06 – 55 74 52 95
Fax	088 – 797 4006
Uitgevoerd door	Ir. H.J. van der Graaf (RWS) Ir. S.E. van Manen (RWS)
Gereviewd door	
Datum	10 januari 2013
Status	Definitief
Versienummer	0.7

Inhoud

1 Inleiding

Rijkswaterstaat is bezig om de kwaliteit van haar systemen met behulp van RAMS eigenschappen te specificeren en te beheren, waarbij RAMS staat voor betrouwbaarheid (Reliability), beschikbaarheid (Availability), onderhoudbaarheid (Maintainability) en veiligheid (Safety). Centraal in deze aanpak van risicogestuurd bouwen en beheren is de kwantitatieve risicoanalyse.

In toenemende mate is (complexe) software onderdeel van de systemen die Rijkswaterstaat laat bouwen en beheert. En ook software blijkt in de praktijk te kunnen falen en moet dus meegenomen worden in de kwantitatieve risicoanalyse.

In opdracht van Rijkswaterstaat heeft een consortium een methode ontwikkeld die zowel richtlijnen geeft voor het modelleren van softwarefalen in foutenbomen als het schatten van de faalkans van een taakuitvoering door een softwaremodule. Deze methode is TOPAAS genoemd en uitgebreid geëvalueerd in [2] en gerapporteerd in [1].

Kern van TOPAAS is dat software in modulen kan worden opgedeeld en dat het (mogelijk) falen van deze modulen in een foutenboom als basisgebeurtenissen kunnen worden opgenomen.

Het schatten van de faalkans van een softwaremodule gebeurt op basis van expert opinion, waarbij het Bayesiaanse gedachtegoed wordt gevolgd. Deze schatting wordt vervolgens gebruikt in een parametermodel. Dit model genereert dan een schatting van de faalkans van de softwaremodule per aanspraak.

Dit document is een korte handleiding voor het toepassen van TOPAAS. Dat gebeurt, zoals aangegeven, op basis van expert opinion in een gezamenlijke sessie. Hierbij is de aanwezigheid van de makers van de software, van ten minste een onafhankelijke IT expert en een of twee procesbegeleiders noodzakelijk. Specifieke kennis van Bayesiaanse kansrekening is van veel minder belang.

Deze rapportage is als volgt opgedeeld in de volgende onderdelen:

- Hoofdstuk 2 richt zich op de definitie van falen en faalkansen. Het legt het verschil uit tussen het traditionele technische falen van elektromechanische componenten, menselijk falen en software falen;
- Hoofdstuk 3 behandelt de opzet van de expertsessie waarin softwaremodules m.b.t. de TOPAAS methode worden beoordeeld;
- Hoofdstuk 4 gaat in op het parametermodel dat wordt gehanteerd in de TOPAAS methode. Hier worden de 15 aspecten van het model toegelicht en de karakteristieke uitkomsten per aspect gegeven. Tevens wordt toegelicht op welke wijze men van de uitkomsten van het parametermodel tot een faalkans per vraag voor een softwaremodule komt.

- Hoofdstuk 5 behandelt de specifieke invulling van de 15 aspecten en zijn vastgestelde parameters. Hierin wordt ook een toelichting gegeven op de definities en begrippen achter de aspecten;
- Hoofdstuk 6 beschrijft een voorbeeld van toepassingen van de methode.

2 Kwantitatieve modellering van software falen

2.1 Kwantitatieve risicoanalyse: de foutenboom

Een kwantitatieve risicoanalyse maakt gebruik van grafische representaties van de mogelijke ongewenste gebeurtenissen. Heel algemeen gesteld, wordt eerst het systeem gedefinieerd en vervolgens worden op een gestructureerde manier alle mogelijke manieren van falen vastgesteld. Daarmee wordt, soms via een gebeurtenissenboom (indien herstel een belangrijke rol speelt), een foutenboom gegenereerd. In bijzondere gevallen zelfs meerdere foutenbomen.

Deze foutenbomen zijn de basis voor het kwantificeren van de “ongewenste topgebeurtenis”: de gebeurtenis waarbij het systeem als gefaald wordt beschouwd. Deze gebeurtenis wordt topgebeurtenis genoemd omdat hij zich (grafisch) boven aan de boom bevindt. Een bekend voorbeeld van een ongewenste topgebeurtenis is het “niet-sluiten” van een stormvloedkering, terwijl dat wel gewenst is.

De kans op ongewenste topgebeurtenis, welke de betrouwbaarheid of de beschikbaarheid van het systeem representeert, wordt berekend op basis van de kans op falen van onderdelen van het systeem, van componenten. Indien de faalkans van zo’n component invoer voor de foutenboom is (en dus niet wordt berekend uit deelcomponenten) wordt het falen van zo’n component een “basisgebeurtenis” genoemd.

Softwaremodulen kunnen beschouwd worden als componenten waarvan de faalkans middel TOPAAS wordt geschat en welke als invoer kunnen dienen voor een foutenboom. Het falen van de module wordt dan gemodelleerd als een basisgebeurtenis. Uiteraard moet falen van de software, mogelijk in combinatie met andere gebeurtenissen, leiden tot falen van het systeem.

2.2 Decomponeren van de software

Om te komen tot een betrouwbare schatting is het noodzakelijk om een complex software systeem te decomponeren tot een verzameling software modules en vervolgens de kans per module af te schatten.

Een software module wordt gedefinieerd op basis van een gebalanceerd abstractieniveau door de taken die de software moet uitvoeren als basis te nemen. Een software module wordt aldus als volgt gedefinieerd:

Een **Softwaremodule** is een taak of een consistente verzameling taken die door een afgebakende verzameling logische regels programmacode (of grafisch equivalent) wordt gerealiseerd, waarbij geldt dat:

- Een duidelijke afbakening te onderkennen is ten opzichte van andere stukken code én er een onderkende functionele doelstelling in het totaal van het systeem aanwezig is;
- Te verifiëren kwaliteitseigenschappen en taakuitvoering aanwezig zijn.

De verzameling logische regels wordt in zeer brede zin beschouwd:
- een softwaremodule kan een specifieke applicatie zijn
- een softwaremodule kan alle software op een dedicated hardware component (PLC) zijn,
- een softwaremodule kan een proces of executable op een mainframe zijn
- een softwaremodule kan een collectie samenwerkende processen in een systeem zijn.

Kern is dat een softwaremodule een taak uitvoert, onderscheidbaar is en enige mate van onafhankelijkheid is van de overige modulen.

Het falen van een softwaremodule is als volgt gedefinieerd:

> **Falen** is de (te lange) afwezigheid van gewenst gedrag, of het uitvoeren van verkeerd gedrag, van een softwaremodule dat noodzakelijk is om zijn taak binnen de missie te volbrengen.

Expliciet valt hieronder:
1. Geen gedrag vertonen
2. Het verkeerde gedrag vertonen
3. Gedrag niet op het juiste tijdstip vertonen

Nu het falen van een softwaremodule is benoemd, kan men ook de faalkans van een software module benoemen:

> De **Faalkans** is kans dat een software module faalt op basis van een vraag tot een gewenste taakuitvoering

2.3 Bouwtekeningen van systemen bestaande uit software

Als men een analyse van software wil maken, moet men een set “bouwtekeningen” hebben om als basis te dienen voor de ontwikkeling van de foutenboom. Net als bij werktuigbouwkundige en elektrotechnische bouwtekeningen moet de architectuur van het systeem een consistent beeld scheppen van de samenhang van alle softwaremodulen die bij elke missie betrokken zijn. Een belangrijke vraag is bijvoorbeeld welke kritieke ketens van softwaremodulen het gedrag van het object bepalen.

De bouwtekeningen moeten dus inzage geven in hoe een missie is opgebouwd uit taken die door componenten worden uitgevoerd. Hiermee kan men de stap naar een foutenboom maken. In deze foutenboom is de specifieke falende taakuitvoering van de component erg belangrijk: dat wordt de basisgebeurtenis.

Een analyse van de architectuur behoort dus antwoord te geven op de volgende vragen die in vervolgstappen gesteld gaan worden:
1. Welke softwaremodulen onderscheiden/identificeren we?
2. Hoe is een missie softwarematig uit deze softwaremodulen opgebouwd, hoe werken de componenten samen om het beoogde resultaat te bereiken (hoe communiceren ze onderling, zijn er parallelle en of

redundante processen, zijn er watchdogs, zijn er on-demand processen)?
3. Van welke hardware is een softwaremodule afhankelijk?
4. Met welke andere processen deelt de softwaremodule de hardware
5. Welke bibliotheken gebruikt de softwaremodule en wat is de kwaliteit?

In de ICT-sector groeit consensus over de beste manier van systeemontwerpen. In de IEEE 1471 [3] zijn de randvoorwaarden voor systeemontwerpen vastgelegd. De meest gebruikte werkwijze van ontwerpen die aan de IEEE 1471 eisen invulling geeft is de 4+1 methode van Kruchten [4] Deze methode is ook in gebruik in het analyseren van luchtverkeersleidingsystemen ten behoeve van FAA certificatie.

Een software architectuur volgens 4+1 bevat de volgende views:
- **Logische view**, die een beschrijving geeft van de functionele onderdelen van een systeem en de functionaliteit voor de eindgebruiker. Ook geeft deze view de randvoorwaarden voor goed functioneren van de software en zijn onderdelen
- **Development view**, die een beschrijving geeft van het systeem vanuit de ontwikkelaar en houdt zich bezig met de organisatie van de software gedurende de ontwikkeling in termen van pakketten broncode, gebruikte standaardbibliotheken van code, compilers, linkers en de onderlinge afhankelijkheden daartussen
- **Process view**, die een beschrijving geeft van alle actieve processen gedurende executie, inclusief alle benodigde communicatie, synchronisatie, redundantie en concurrency-control
- **Physical view**, die een beschrijving geeft van de hardware-infrastructuur (PLC's, servers en netwerken) en de allocatie van de run-time onderdelen daarop, vanuit het perspectief van de system engineer. Hier moeten dus alle individuele hardware componenten genoemd zijn die ook in de foutenboom worden onderscheiden
- **Use-case view / Scenario's**, dat, door middel van een kleine set voorbeelden (use-cases) of scenario's inzicht geeft in de samenhang tussen de verschillende componenten. Hierbij wordt expliciet niet alleen het primaire scenario beschreven, maar ook de alternatieve scenario's die, bijvoorbeeld, kunnen ontstaan ten gevolge van redundantie of door monitoring met restart functie (watchdog resets). Ten minste moet deze view alle uit te voeren taken weergeven die voor de foutenboom van belang zijn.

Met deze views kan een elementair beeld worden opgebouwd hoe het systeem werkt, vergelijkbaar aan een hydraulische, elektrische of mechanische bouwtekening. Deze bouwtekeningen zouden óf door leveranciers opgeleverd moeten worden bij oplevering óf door een software analist in retrospect moeten worden vastgesteld op basis van alle andere aanwezige documentatie.

3 Benodigde expertise en aanpak van een faalkans sessie

3.1 Benodigde expertise

Het uitvoeren van een analyse op software gebeurt door middel van expert opinion. Hiertoe moet een team worden samengesteld waarin alle benodigde expertise aanwezig is. De groep bestaat minimaal uit:

- een onafhankelijke IT expert die het toepassen van de TOPAAS methode beheerst;
- de maker(s) van de software die betrokken zijn geweest bij het ontwikkelproces, het ontwerpproces, het programmeren en het testen;
- een of twee procesbegeleiders die kennis van TOPAAS hebben en de resultaten van het proces kunnen vastleggen.

Specifieke kennis van de Bayesiaanse kansrekening is van veel minder belang.

3.2 Aanpak faalkans sessie

Een goede voorbereiding van faalkans sessie(s) is belangrijk. Er dient vooraf aan de onafhankelijke IT expert volledig inzicht te worden verschaft over de te beoordelen software. Daarbij gaat het om:

- info over het ontwikkelproces;
- de omgang van wijzigingen in het ontwerp;
- de modulaire opbouw van de software;
- de geprogrammeerde taakuitvoering per module;
- het belang van de module in het functioneren van het object;
- de omvang van de modulen;
- de onderlinge samenhang van de modulen.

Van de makers van de software wordt verwacht dat zij voorafgaand aan de expertsessie kennis nemen van de TOPAAS methode.

De duur van een expert opinion sessie wordt in belangrijke mate bepaald door de omvang van de software (het aantal modules dat moet worden geanalyseerd) maar ook door de wijze waarop door de makers van de software inzage wordt gegeven in het ontwikkel- en maakproces van de software. In feite moeten alle vragen in de sessie onderbouwd worden beantwoord.

Voor een goed verloop van een sessie moeten ruim de tijd worden genomen en moet men niet gestoord worden door andere afspraken en storende invloeden. Het gebruik van computer, presentatieschermen, flip-overs en ruimte voor het uitleggen van tekeningen en schema's wordt aanbevolen. Indien aan deze voorwaarden kan worden voldaan is op basis van ervaring gebleken dat het mogelijk is twee dominante modulen in een dagsessie te behandelen. Bij meerdere modulen zal een versnelling optreden indien blijkt dat van een aantal generieke aspecten voor alle modulen het zelfde geldt.

Voor de bepaling van de bijdrage aan de faalkans van de 15 aspecten waarop de software wordt beoordeeld wordt aanbevolen gebruik te maken

van het spreadsheet “TOPAAS lege scoretabel v2.0 30-5-2011.xls”, maar het resultaat kan ook eenvoudig handmatig worden berekend.

4 Schatting van faalkans taakuitvoering

4.1 Beschrijving parametermodel op hoofdlijnen

Om tot een schatting van de faalkans van een module (taakuitvoering) te komen is, in navolging van het Bayesiaans gedachtegoed, gebruik gemaakt van een gedeeld expert oordeel van een groep experts. Om tot een herhaalbaar en praktisch bruikbaar proces te komen is deze inschatting gemodelleerd in een parameter model. De gebruikte parametrisering in het model is dus een reflectie van de expert opinions. Hiermee wordt de afhankelijkheid van een specifieke groep experts verminderd en wordt de controleerbaarheid vergroot. Met andere woorden: het parametermodel levert een benadering van een expert-group op, om zodoende de faalkans te schatten.

Het TOPAAS model is dus een parametermodel zoals op het OPSCHEP-model [5] dat is voor menselijk falen. De 15, voor de kwaliteit van de software belangrijkste eigenschappen zijn geparametriseerd. Deze parameters worden in deze handleiding ook wel factors of aspecten genoemd.

Het parametermodel van TOPAAS gaat uit van een conservatieve initiële faalkans Op basis van beschikbare informatie over de softwaremodule wordt deze initiële kans per aspect bijgesteld: meestal gereduceerd en soms vergroot. Hierbij wordt expliciet de nadruk gelegd op het beschikbaar zijn van informatie, alleen als die aanwezig is volgt een aanpassing van de geschatte faalkans. Als van een aspect geen informatie beschikbaar is wordt de faalkans niet aangepast. Dit kan omdat de initiële faalkans voldoende conservatief is ingesteld (zie §5.1 “Initiële faalkans”). Als men van een aspect van de software geen informatie heeft, kan men dat aspect gewoon overslaan.

Per aspect is voor elk stuk relevante informatie een schaal bepaald waarop de softwaremodule ingedeeld (“gescoord”) moet worden, resulterend in een vermenigvuldigingsfactor. De vermenigvuldigingsfactor kan groter of kleiner dan één zijn. Deze vermenigvuldigingsfactor is bepaald op basis van een, door de experts veronderstelde correlatie met de faalkans. De uiteindelijke faalkans is dan de initiële faalkans vermenigvuldigd met alle vermenigvuldigingsfactoren die zijn vastgesteld.

De vorm van het parametermodel is daarmee:

$$P = P_B * F_i * F_2 * \ldots * F_n$$

waarbij P_B de initiele faalkans is en F_i de vermenigvuldigingsfactoren.

In het TOPAAS model wordt de initiële faalkans (P_B) op 1 gezet (zie §5.1 ”Initiële faalkans”). Alle F_i’s zijn dan vermenigvuldigingsfactoren die een veronderstelde invloed hebben op de faalkans.

In de praktijk rekenen we bij TOPAAS met de logaritme van de bovenstaande formule:

$$\log(P) = \log(F_1) + \log(F_2) + \cdots + \log(F_n) = \sum_{i=1}^{n} B_i$$

waarbij TOPAAS per aspect de factoren B_i geeft. Hiermee wordt de vermenigvuldiging een eenvoudige optelling. Het bepalen van de uiteindelijke faalkans gebeurt dan met:

$$P = 10^{\sum_{i=1}^{n} B_i}$$

De expertgroep heeft nog twee extra voorwaarden aan de kansschatting gesteld:

1. het eindresultaat wordt naar boven (ongunstig) afgerond op hele machten van 10
2. de faalkans van een module kan nooit kleiner zijn dan 10-5 (per vraag).

5 Specifieke invulling parametermodel

5.1 Initiële faalkans

Als initiële faalkans is voor 1 gekozen, omdat dit de meest conservatieve waarde is die men kan kiezen. Bij vergelijkbare modellen, bijvoorbeeld het OPSCHEP model, wordt gekozen voor een gemiddelde faalkans. Een gemiddelde is bij software echter onbepaald, daarom is voor een zeer conservatieve benadering gekozen. De keuze voor een initiële faalkans van 1 komt de onderhoudbaarheid van de methode ten goede, als op een later moment blijkt dat extra factoren toegevoegd moeten worden, zal de herijking van bestaande analyses minder vaak nodig blijken.

Gevolg is dat, als men absoluut niets weet van de softwaremodule, deze met een faalkans 1 in de foutenboom wordt opgenomen. Gezien de afwezigheid van enige informatie en het niet uitgevoerd hebben van enige simulaties (een parameter die een afnemer zelf kan beïnvloeden) lijkt dit geen onterechte conservatieve aanname.

5.2 Invulling methode

Het schatten van de faalkans van de softwaremodule gebeurt via een score op de volgende dimensies:

a) Totstandkomingproces
b) Product
c) Requirements traceability/verifieerbaarheid
d) Testen
e) Executieomgeving/gebruik

De aspecten die in deze dimensies een rol spelen zijn begrippen die voornamelijk bekend zijn in de ICT-wereld. Het schatten waar de grenzen liggen tussen de genoemde niveaus per aspect is niet altijd SMART. De kennis van een ICT-expert (software expert) is noodzakelijk bij het goed schatten van het juiste niveau. Ook een onafhankelijke blik op de aspecten is van belang. De softwaremaker zelf zal vaak een te rooskeurige blik hebben van zijn eigen proces of product. Daarom wordt sterk aanbevolen om bij de beoordeling van de software een onafhankelijke ICT-expert te betrekken.

5.2.1 Totstandkomingproces

1 Het ontwikkelproces voldoet aan een van de SIL's van de IEC61508 [6]		
1	Onbekend, het ontwikkelproces voldoet niet aantoonbaar aan een SIL niveau	0
2	Ontwikkelproces voldoet aantoonbaar niet aan een SIL niveau door het gebruik van Not Recommended practices	½
3	Ontwikkelproces voldoet aantoonbaar aan SIL-1 niveau	-½
4	Ontwikkelproces voldoet aantoonbaar aan SIL-2 niveau	-1
5	Ontwikkelproces voldoet aantoonbaar aan SIL-3 niveau	-2
6	Ontwikkelproces voldoet aantoonbaar aan SIL-4 niveau	-3

Toelichting: De reductie van de (initële) faalkans lijkt wat gering indien bij de totstandkoming een SIL (1-4) is bereikt (IEC61508 [6]). Maar het is ondenkbaar dat een SIL bereikt wordt zonder dat de overige aspecten de faalkans verder reduceren.

Optie 2 is van toepassing als een specifiek Safety Integrity Level wordt vereist voor de betreffende module, en daar door de leverancier niet aan wordt voldaan vanwege het toepassen van een of meerdere 'not recommended practices' (een situatie die in praktijk dus niet of nauwelijks zal voorkomen).

2 Gebruik van Inspecties			
		Normaal	**SIL3/SIL4**
1	Onbekend	0	NVT
2	Geen inspecties uitgevoerd	⅓	NVT
3	Aantoonbaar inspecties op ontwerpen en code uitgevoerd	0	⅓
4	Aantoonbaar Fagan inspecties uitgevoerd op alle ontwerp en testdocumenten	-½	0

Toelichting: De eerste 2 opties zijn niet van toepassing voor SIL 3 of 4, omdat de IEC61508 inspecties verplicht stelt als men een SIL3/4 systeem realiseert. Onder SIL3/4 worden Fagan inspecties aanbevolen, zie Bijlage A, "Het Fagan-inspectieproces".

3 Hoeveelheid wijzigingen ten opzichte originele ontwerp/eisenpakket		
1	Onbekend	0
2	Zeer frequente of enkele fundamentele wijzigingen	⅔
3	Weinig wijzigingen, met zeer geringe impact	0
4	Geen wijzigingen	-⅓

Toelichting: Onder zeer frequente wijzigingen wordt een meer dan normaal verwacht aantal wijzigingen verstaan. Bij fundamentele wijzigingen moet in de eerste plaats worden gedacht aan functionele wijzigingen in de requirements of specificaties na afsluiting van de betreffende ontwerpfase of wijzigingen in het technisch ontwerp tijdens of na realisatie of constructiefase (bijv. het verschuiven van de modulegrenzen door opknippen of samenvoegen van modulen). Indien voor de optie 3 (Weinig wijzigingen, met zeer geringe impact) of 4 (Geen wijzigingen) wordt gekozen dient een versienummering en logboeken aanwezig te zijn.

4 Cultuur en samenwerking		
1	Onbekend	0
2	Op regels gebaseerde organisatie	⅓
3	Doelgerichte organisatie	0
4	Zelflerende organisatie	-½

Toelichting: De typering van cultuur en samenwerking wordt volgens IAEA-TECDOC-1329 als volgt gedefinieerd:

Type organisatie Kenmerken	**Op regels gebaseerde organisatie**	**Doelgerichte organisatie**	**Zelflerende organisatie**
Kijk op fouten	Verwijten aan personeel i.p.v. luisterend en lerend	Fouten leiden tot meer controles en training	Fouten ziet men als gelegenheden om te leren en te verbeteren
Tijd focus	Korte termijn is het allerbelangrijkste	Mensen worden beloond voor het overtreffen van doelen, los van de gevolgen op langere termijn	Korte termijn performance wordt geanalyseerd om langere termijn te verbeteren
Rol van managers	Managers stellen regels vast en pressen werknemers de gestelde doelen te bereiken	Managers gebruiken technieken zoals “Management by Objectives”	Managers coachen mensen ter verbetering van hun performance en ondersteunen samenwerking
Omgaan met conflicten	Conflicten worden zelden opgelost en groepscompetitie blijft aanwezig	Conflicten worden ontmoedigd in naam van het teamwork	Conflicten worden opgelost door oplossingen die voor beide partijen acceptabel zijn
Kijk op mensen	Mensen zijn componenten in een systeem	Besef dat het gedrag van mensen invloed heeft op hun prestaties	Mensen worden gerespecteerd en gewaardeerd voor hun bijdrage

5 Opleidingsniveau en ervaring ontwikkelaars		
1	Onbekend, of enige kennis en ervaring van het specifieke domein	0
2	Geen kennis en ervaring van het specifieke domein	1
3	Weinig kennis en ervaring van het specifieke domein	½
4	Gewenste kennis met systeemontwikkeling voor het specifieke domein (onbewust bekwaam)	0
5	Aantoonbaar uitstekende kennis en veel ervaring van het specifieke domein	-½

Toelichting: Er wordt uitgegaan van uitstekende kennis en ervaring van systeemontwikkeling. Met “Weinig kennis en ervaring” wordt gedacht aan (een of meer) projecten, langer dan drie jaar geleden. Met “uitstekende kennis en ervaring” wordt bedoeld dat er een of meer projecten van meer recente datum aangetoond kunnen worden. Met “domein” wordt expliciet een objectdomein bedoeld: een systeem voor een

stormvloedkering, voor een sluis, voor een beweegbare brug, voor een tunnel, etc.

6 Samenwerking met Opdrachtgever		
1	Onbekend	0
2	Niet nauw betrokken opdrachtgever met weinig IT kennis, sterk financieel gedreven opdrachtgever	½
3	Zijdelings betrokken opdrachtgever met matige IT kennis	0
4	Sterk betrokken opdrachtgever met voldoende kennis, open dialoog waarbij de opdrachtgever bereid is om overall architectuur te wijzigen als dat de software betrouwbaarheid ten goed komt. Er is sprake van een systems engineering approach voor de overall ontwikkeling van het systeem.	-½

Geen nadere toelichting.

5.2.2 Producteigenschappen

7 Complexiteit beslislogica		
1	Onbekend	0
2	Beslislogica is zeer complex (bevat veel vertakkingen en uitzonderingen), McCabe index groter dan 60	½
3	Beslislogica matig complex (bevat enkele uitzonderingssituaties), McCabe index tussen 30 en 60	0
4	Beslislogica is redelijk eenvoudig (bevat enkele zeer geïsoleerde uitzonderingssituaties), McCabe index tussen 10 en 30.	-⅓
5	Beslislogica en foutherkenning zijn erg eenvoudig, McCabe Index kleiner dan 10	-½

Toelichting: Beslislogica is cruciaal: men kan kijken naar de McCabe complexiteit van de broncode, maar ook naar het extern observeerbaar gedrag van de component en op basis daarvan een inschatting maken. Meest geaccepteerde vorm van het meten van complexiteit van software is de McCabe index. Alhoewel er formele wiskundige definities zijn, is de meest eenvoudige definitie:

$$McCabe\ Index = 1 + D$$

met D = het aantal beslispunten (if/case statements, while statements, switch/case statements). Deze definitie kan zowel toegepast worden op broncode (tellen van de if/case, while en switch/case statements), als op beschreven beslislogica van de applicatie.

Daarbij is het van belang dat alle code van de beschouwde module wordt meegenomen. Wanneer de module voor een belangrijk deel bestaat uit code die niet direct te relateren is aan het top event van de FTA of de missie van het systeem, dient de module verdere decompositie te ondergaan en

moeten de afzonderlijke componenten daaruit in een nieuwe FTA worden ondergebracht.

8 Omvang softwaremodule (Lines of code)		
1	Onbekend	0
2	Meer dan 50.000	½
3	Tussen 10.000 en 50.000	⅓
4	Tussen 5.000 en 10.000	0
5	Tussen 1.000 en 5.000	-⅓
6	Minder dan 1000	-½

Toelichting: Bij een “Component of the Shelf” (COTS) is het vaak onmogelijk om het aantal regels code vast te stellen. In dat geval is het toch vaak mogelijk om een indruk van de omvang van de software te krijgen met behulp van een functiepuntenanalyse [7].
Indien met behulp van een grafische programmeeromgeving wordt gewerkt, wordt één grafisch element als één Line of Code opgevat.

Bij gebruik van C is de volgende vuistregel van toepassing: Elke ontwerppagina resulteert in circa 200 tot 250 regels C-code (exclusief de zuiver inleidende pagina’s). Het verkregen gemiddelde per pagina is meerdere malen geëvalueerd bij verschillende projecten.

9 Helderheid gebruikte architectuurconcepten		
1	Onbekend	0
2	Geen heldere afbakening taakuitvoering modulen in ontwerp benoemd	½
3	Wel taakuitvoering op hoofdlijnen benoemd, maar geen navolging gegeven in ontwikkeling	⅓
4	Er is een scheiding van taakuitvoering tussen modulen beschreven, welke het principe van “maximale cohesie en minimale koppeling” respecteert, maar deze is passief bewaakt tijdens het ontwikkelproces	0
5	Er is een scherpe scheiding van taakuitvoering tussen modulen beschreven op basis van geldende documenten, welke het principe van “maximale cohesie en minimale koppeling” respecteert, en deze is aantoonbaar actief bewaakt tijdens het ontwikkelproces	-½

De actieve bewaking van het principe van “maximale cohesie en minimale koppeling” houdt in dat de methode van systeemontwikkeling dit principe afdwingt en er tijdens de verschillende ontwikkelfasen dus geen onopgemerkte afwijkingen in het ontwerp kunnen worden aangebracht. Wanneer alleen sprake is van standaards en richtlijnen op dit gebied spreken we van “passieve” bewaking.

Bij beschouwing van modules op een hoog aggregatie niveau (bijv. systeem componenten als SCADA) komt de begrenzing overeen met de grenzen van de hardware of COTS module. In dergelijke gevallen is er geen bewuste ontwerpkeuze met

betrekking tot de cohesie en koppeling aan voorafgegaan en is optie 4 (scheiding maar met passieve bewaking) van toepassing; hetgeen geen beïnvloeding van de faalkans met zich meebrengt

10 Gebruik van een gecertificeerde compiler			
		Normaal	**SIL3/SIL 4**
1	Onbekend	0	NVT
2	Gebruik van een willekeurige compiler	⅓	NVT
3	Gebruik van een compiler waar de ontwikkelaar langdurige ervaring mee heeft	0	⅓
4	Gebruik van een certified compiler in combinatie met een gevalideerde safe subset	-½	0
5	Gebruik van een certified compiler in combinatie met een gevalideerde safe subset met bijbehorende kalibratiesets en testprotocol om compiler te ijken/testen, wat voor elke nieuwe versie van de compiler structureel gebeurt	-⅔	-⅓

Toelichting: Er zijn maar weinig **gecertificeerde** compilers. Zo zijn, bijvoorbeeld, ook de compilers die Microsoft levert bij Windows geen **gecertificeerde** compilers. Bij ontwikkeling op SIL 3/4 wordt, vanuit de IEC61508, een **gecertificeerde** compiler verplicht gesteld.
De IEC61508, deel 7 zegt specifiek over het gecertificeerd zijn: "*The certification of a tool will generally be carried out by an independent, often national, body, against independently set criteria, typically national or international standards. (…). To date, only compilers (translators) are regularly subject to certification procedures; these are laid down by national certification bodies and the exercise compilers (translators) against international standards such as those for Ada and Pascal".*

5.2.3 Requirements traceability/verifieerbaarheid

11 Traceerbaarheid van requirements door het proces heen			
		Normaal	**SIL3/SIL4**
1	Onbekend	0	NVT
2	Geen traceerbaarheid	⅓	NVT
3	Aantoonbaar traceerbaar naar testscripts	0	NVT
4	Aantoonbaar traceerbaar naar architectuur en testen	-⅓	⅓
5	Aantoonbaar traceerbaar van veiligheidskritische eisen tot en met de code en individuele testen	-⅔	0
6	Aantoonbare volledige traceerbaarheid tot en met de code	-1	-⅓
7	Aantoonbaar wiskundig/logisch bewezen correcte traceerbaarheid	-2	-½

Toelichting: Dat een software module “doet wat hij moet doen” betekent dat de opgestelde gebruikerseisen allemaal gerealiseerd zijn en het liefst aantoonbaar. De beste manier om dat te bewerkstelligen is door in het software ontwikkelproces, stapsgewijs de gebruikerseisen te vertalen naar testscripts en technische systeemeisen. Vervolgens worden software modules aangewezen die de systeemeisen waar gaan maken. Op deze manier worden gebruikerseisen (de requirements) traceerbaar tot en met de code.

Wiskundig correct bewezen traceerbaarheid (optie 7) houdt in dat er model checking toegepast is of een gedragsequivalentie aangetoond is tussen de implementatie en een correcte specificatie. Dit is vooralsnog alleen voor kleinschalige systemen te realiseren. Als de links tussen alle gebruikerseisen, tests, onderliggende systeemeisen en uitvoerende code duidelijk zijn dan is de traceerbaarheid in de software module volledig (optie 6). Dit geldt misschien alleen voor veiligheidskritische eisen (optie 5) of alleen gedeeltelijk (optie 4) of tot tests (optie 3).

Voor modules waarbij geen sprake is van veiligheidskritische functies of eisen, is optie 5 niet van toepassing.

5.2.4 Testen

12 Testtechnieken en dekkingsgraad			
		Normaal	**SIL3/SIL4**
1	Onbekend	0	NVT
2	Geen gedocumenteerde tests uitgevoerd	0	NVT
3	Wel testen gedocumenteerd, geen formele testtechnieken gehanteerd; dekkingsgraad onbekend	-⅓	NVT
4	Formele testtechniek(en) gehanteerd met lage dekkingsgraad	-½	⅔
5	Formele testtechniek(en) gehanteerd met gemiddelde dekkingsgraad	-⅔	½
6	Formele testtechniek(en) gehanteerd met hoge dekkingsgraad	-1	0
7	Formele testtechniek(en) gehanteerd met aantoonbare (gemeten) hoge dekkingsgraad	-1⅓	-⅓

Toelichting: Bij het formeel testen worden er technieken toegepast uit wiskunde en logica. Het startpunt is een geverifieerd model van het softwareontwerp, of de software zelf als die geannoteerd is met formele specificaties. Daaruit worden automatisch tests gegenereerd die een gewenste requirement testen met een gecontroleerde dekkingsgraad. Het formeel testen proces maakt het verband tussen requirement en test case heel duidelijk en is daardoor van belangrijke waarde voor het aantonen van software betrouwbaarheid. Een paar tools die model-based testgeneratie ondersteunen zijn T-VEC,

Conformiq, Reactis, Unitesk. Een academische tool die intensief werd gebruikt in de telecommunicatiewereld is TGV, met TorX als opvolger.

In modules waarin geen sprake is van geverifieerde modellen van het softwareontwerp (zoals administratieve of andere niet veiligheidskritische toepassingen) mogen ook beslissingstabellen, data-flow- en proces-cyclus-testen als formele testspecificatietechniek worden beschouwd.

De dekkingsgraad van testgevallen kan op verschillende manieren worden uitgedrukt:

- afhankelijk van gebruikte specificatietechniek:
  - beslissingstabellen op basis van statement-, condition- of multiple condition coverage;
  - procescycli (aan de hand van beslispunten) op basis van 'testmaat 1', '- 2' of '-n';
  - combinatie van equivalentieklassen op basis van pairwise of triplewise; etc..
- als percentage van doorlopen paden tijdens testtraject t.o.v. theoretisch mogelijke paden m.b.v. tools zoals McCabe (vereist dus inzet van tools tijdens testtraject en beschikbaarheid van source code daarbij).
  De dekkingsgraad is in dit geval:
  - hoog: 90% van de mogelijk door te lopen paden;
  - gemiddeld: 50% van de mogelijk door te lopen paden;
  - laag: 10% van de mogelijk door te lopen paden.
- als percentage van doorlopen combinaties van input variabelen ten opzichte van theoretisch mogelijke combinaties van input variabelen.
  De dekkinggraad is in dit geval:
  - hoog: 90% van de theoretisch mogelijke input;
  - gemiddeld: 50% van de theoretisch mogelijke input;
  - laag: 10% van de theoretisch mogelijke input.

5.2.5 Executieomgeving/gebruik

13 Multiprocesomgeving		
1	Onbekend	0
2	Meerdere applicaties draaien parallel op één OS in een netwerkomgeving	½
3	Meerdere applicaties op één stuk hardware	⅓
4	Één proces op een dedicated OS met een dedicated CPU	0
5	Dedicated CPU en memory op geen of een triviaal OS	-⅓

Geen nadere toelichting.

14 Aanwezigheid representatieve velddata gedurende taakuitvoering		
	Normaal	**SIL3/SIL4**

1	Onbekend	0	NVT
2	Geen velddata aanwezig,	⅓	⅓
3	Beperkte gegevens aanwezig en geanalyseerd uit periode tijdens uitvoeren van de functie van de module	0	0
4	Significante hoeveelheid gegevens aanwezig en gebruikt uit periode tijdens uitvoeren van de functie van de module	-1	-⅓
5	Veel representatieve velddata aanwezig en gebruikt van identieke of sterk vergelijkbare toepassingen	-2	-½

Toelichting: Het betreft hier velddata in de statistische zin van de klassieke faalkans-berekening. Bij gebruik van een relatief simpele COTS bouwsteen van Siemens of ABB is er weliswaar veel betrouwbare data beschikbaar over hoe het algemene product zich gedraagt in het veld. De vraag is echter of de algemene toepassing in het veld REPRESENTIEF is voor de toepassing in de analyse.

Bij keuze 3 wordt met “beperkte gegevens” bedoeld dat de module erg weinig gebruikt wordt, minder dan, ordegrootte, 5 keer per maand.
Bij keuze 4 wordt met “significante hoeveelheid gegevens” bedoeld dat de module beperkt wordt gebruikt, ten minste, ordegrootte, 5 keer per maand.
Bij keuze 5 wordt met “veel representatieve velddata” bedoeld dat de module veel wordt gebruikt, ten minste, ordegrootte, 5 keer per dag.

15 Monitoring		
1	Onbekend	0
2	Geen aanwezig	⅓
3	Weinig/kort gemonitord gedurende taakuitvoering	0
4	Langdurige monitoring, maar infrequente taakuitvoering	-⅓
5	Langdurige/frequente monitoring tijdens taakuitvoering	-½

Toelichting: De monitoring is gericht op de taakuitvoering van de module specifiek in de geanalyseerde toepassing. Daarbij kunnen ook andere omgevingen waarin de module draait betrokken worden.

Bij keuze 3 wordt met “Weinig/kort gemonitord” bedoeld dat de module beperkt wordt gebruikt, hooguit, ordegrootte, 1 keer per maand.

Bij keuze 4 wordt met “Langdurige monitoring, maar infrequente taakuitvoering” bedoeld dat de module ten minste, ordegrootte, 1 keer per maand wordt gebruikt en maar hooguit 1 keer per jaar wordt aangesproken op de bijzondere situatie waarvoor de software is geschreven.

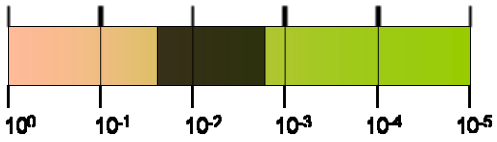
Bij keuze 5 wordt met “Langdurige/frequente monitoring tijdens taakuitvoering” bedoeld dat de module ten minste, ordegrootte, 5 keer per dag wordt gebruikt en ten minste, ordegrootte 5 keer per jaar, wordt aangesproken op de bijzondere situatie waarvoor de software is geschreven.

5.3 Invulling tijdens verschillende fasen van ontwikkeling

Bij de totstandkoming van het TOPAAS model is uitgegaan van het toepassen van de methode na oplevering van het betreffende systeem. In de praktijk blijken opdrachtnemers echter behoefte te hebben aan analyses tijdens de ontwikkeling, op verschillende momenten in het traject. Sommige aspecten en factoren kunnen dan nog niet beoordeeld worden omdat bijvoorbeeld het testtraject nog niet is doorlopen en dus geen uitspraak gedaan kan worden over de toegepaste testtechnieken. De faalkans wordt daardoor beïnvloed waardoor een afwijkend beeld kan ontstaan ten opzichte van de werkelijke situatie achteraf.

De volgende richtlijnen moet hiervoor worden gehanteerd:

Vul bij de factoren die nog niet bepaald kunnen worden de waarde in voor de realistisch negatieve situatie en vervolgens, na berekening van het eindresultaat, voor de realistisch positieve situatie (op basis van de geschetste plannen en vooruitzichten). Aldus ontstaat een bandbreedte van de faalkans dit in het eindrapport als volgt wordt weergegeven:



De uiteindelijke score wordt in een finale analyse achteraf bepaald.

5.4 Configureerbare COTS toepassingen

TOPAAS is bedoeld om te kunnen toepassen zowel op maatwerk als op commerciële standaard- of pakketsoftware. In veel gevallen zullen bepaalde factoren voor de laatste categorie niet ingevuld kunnen worden, zoals ‘cultuur en samenwerking’, ‘hoeveelheid wijzigingen’, ‘opleidingsniveau medewerkers’, etc. Dergelijke factoren worden op ONBEKEND gezet en hebben daarmee geen invloed op de faalkans.

Een bijzonder omstandigheid ontstaat wanneer een standaardpakket door middel van parametersettings geconfigureerd kan worden tot de vereiste functionaliteit. In procesbesturing is dat bijvoorbeeld het geval voor een SCADA systeem. Maar in ERP toepassingen geldt dat in nog veel sterkere mate. Configuratie van een pakket als SAP, vergt een volledig ingericht ontwikkeltraject. Het is dan beter het pakket niet als zodanig te beschouwen maar als ontwikkelomgeving.

In dergelijke gevallen hebben alle TOPAAS factoren betrekking op het configuratieproces waarmee de faalkans van het pakket zelf dus buiten

beschouwing wordt gelaten. Dit is acceptabel ervan uitgaande dat de faalkans van het ‘pakket’ vele malen kleiner is dan de faalkans die wordt geïntroduceerd door het configuratieproces, waardoor deze slechts een marginale invloed zal hebben op het totaal.

Wanneer de faalkans van het pakket en die van het configuratie proces beide in beschouwing genomen moeten worden, dan kan het pakket als twee modules in de FTA worden opgenomen waarbij m.b.v. TOPAAS één maal het pakket en één maal het configuratie proces wordt geanalyseerd. Een andere mogelijkheid is een mix van beide beschouwingen, waarbij voor de factoren die voor het pakket ‘onbekend’ zijn, het configuratieproces wordt beoordeeld en omgekeerd. Concreet betekent dit dat voor bijvoorbeeld ‘cultuur en samenwerking’, ‘hoeveelheid wijzigingen’ en ‘opleidingsniveau medewerkers’ naar het configuratieproces wordt gekeken. Terwijl voor ‘aanwezigheid velddata’ en ‘monitoring’ het pakket zelf wordt beoordeeld.

Welke beschouwingswijze de voorkeur heeft hangt af van de omvang van het pakket en het configuratieproces, de complexiteit, de plaats in het systeem e.d., zulks ter beoordeling van de analist.

6 Voorbeeld

6.1 BepaalPeiloverschrijding

6.1.1 Systeemanalyse

Het BOS is het BeslisOndersteund Systeem dat bij de Maeslantkering bepaalt of de kering gesloten dient te worden. Het bepaalt tevens alle tijdstippen die in de sluitprocedure van belang zijn, bijvoorbeeld de waarschuwingen naar het Havenbedrijf Rotterdam, het openen van de dokdeuren, het uitvaren, etc.

In de analyse van het BOS (inclusief script) kan de systeemanalyse functiegedreven zijn door het procedurescript te beschouwen, waar in het geval van een stormconditie een *ProcedureScriptOpdracht* wordt uitgevoerd '*BepaalPeiloverschrijding'*. Op dit moment zijn er drie faalmodi 'verzaken in de bepaling van de Peiloverschrijding' (returnwaarde ONBEPAALD) 'onterecht niet melden van een Peiloverschrijding' en 'onterecht melden van een Peiloverschrijding'.

De peiloverschrijding wordt bepaald op basis van een set beslisregels en een verwachte waterstand bij Rotterdam. De verwachte waterstand wordt geproduceerd door de taken *SRS:ValideerModelInput* en *SOBEK:BepaalWaterstandRotterdam*.

Afhankelijk van incompleetheid van inputdata voor SOBEK wordt de taak *SRS:RepareerInput* uitgevoerd. Condities voor *SRS:RepareerInput* zijn voor deze functie zijn:

- C1: er is een recente (>-3 uur) verwachting van de waterstand bij Hoek van Holland of een redelijk recente verwachting (> - 12 uur) van de waterstand bij Hoek van Holland beschikbaar;
- C2: het astronomisch getij bij Hoek van Holland en een gemeten waterstand bij Hoek van Holland is beschikbaar;
- In geval van C1 wordt *SRS:RepareerInput* niet uitgevoerd;
- In geval van C2 (en niet C1) wordt *SRS:RepareerInput* uitgevoerd;
- In overige gevallen faalt *SRS:RepareerInput* geheel (stopt).

De taken *SRS:ValideerModelInput*, *SOBEK:BepaalWaterstandRotterdam* en *BPO-logica* worden iedere 10 minuten uitgevoerd, waarbij alleen de uitvoering van *BPO-logica* afhankelijk is van de missie.

De taken *SRS:ValideerModelInput* en *SRS:RepareerInput* zijn geïmplementeerd via de component *SRS*, draaiend op de Stratus ftServer, gebruikmakend van BOS-database access. Geïmplementeerd in C/C++ met bijbehorende bibliotheken.

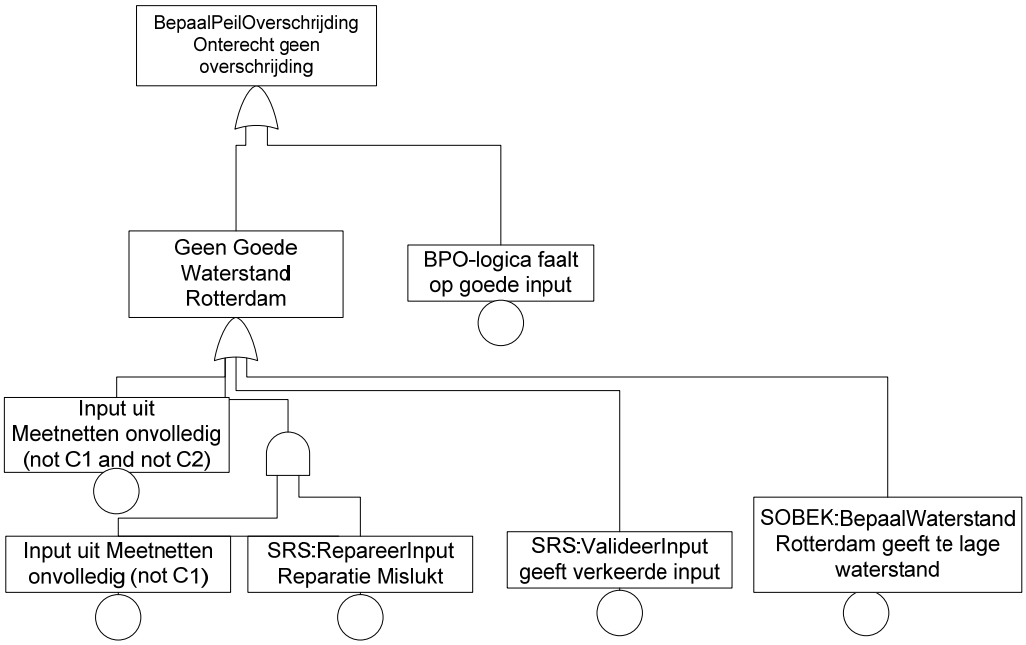
De taak *SOBEK:BepaalWaterstandRotterdam* is geïmplementeerd via de component *SOBEK*, draaiend op de Stratus, geïmplementeerd in Fortran90 gebruikmakend van fortran bibliotheken.

De taak *BPO-logica* is geïmplementeerd in de BOS component *PSI*.

Ondersteunend aan deze taken is het BOS-platform (database, ftServer, OS).

6.1.2 Foutenboom analyse

De foutenboom zou er op dit onderdeel zo uit kunnen zien (er vanuit gaande dat falen van het BOS-platform op een hoger niveau meegenomen wordt). Het niet aanwezig zijn van meetdata is in de huidige boom ook verder uitgewerkt. Dat is hier ook nodig. Ter illustratie is het als basisgebeurtenis opgenomen.



Figuur 1: Foutenboom *BepaalPeilOverschrijding* faalt

6.1.3 Kwantificering

Om kwantificering te kunnen uitvoeren zijn detailgegevens van de taken nodig. Een belangrijke analyse daarin is de hoeveelheid broncode die de taak in beslag neemt en wat de complexiteit van deze code is. De taak *SRS:RepareerInput* wordt door de hele component *SRS* geïmplementeerd. De taak *SRS:ValideerInput* is slechts 10% van de component *SRS* (aanname).

De code voor *BPO-logica* is geïmplementeerd in de BOS component *PSI* in functies *psbepmodelevaluatie* (cyclomatische complexiteit 14 en aantal code statements 116 LOC) en *psbeppeiloverschr* (complexiteit=10 code 162). De component *PSI* is ongeveer 40kLOC, waarvan specialistische taken (zoals *beppeiloverschr*) 10kLOC zijn. Voor de kwantificering wordt 30kLOC genomen met een complexiteit 15 (zijnde de maximale complexiteit in het algemene gedeelte).

Met deze gegevens krijgt men de volgende scoring:

Totstandkomingproces		
1	**Het ontwikkelproces voldoet aan een van de SIL's van de IEC61508** 3 Ontwikkelproces voldoet aantoonbaar aan SIL 1 niveau *Gebouwd door kleine club goede SW engineers met "matige" specifieke procesvastlegging binnen ISO9001 proces.*	-½
2	**Gebruik van Inspecties** 3 Aantoonbaar inspecties op ontwerpen en code uitgevoerd *Reviews uitgevoerd, geen formele inspecties*	0
3	**Hoeveelheid wijzigingen ten opzichte originele ontwerp/eisenpakket** 4 Geen wijzigingen *De main stream is constant*	-⅓
4	**Cultuur en samenwerking** 3 Doelgerichte organisatie *Organisatie met een degelijk trackrecord en een goed georganiseerd intern werkproces*	0
5	**Opleidingsniveau en ervaring ontwikkelaars** 5 Aantoonbaar uitstekende kennis en veel ervaring van het specifieke domein *Gebouwd door kleine club goede SW engineers. Goed in de materie, geen procesvastlegging.*	-½
6	**Samenwerking met Opdrachtgever** 4 Sterk betrokken opdrachtgever met voldoende kennis, open dialoog waarbij de opdrachtgever bereid is om overall architectuur te wijzigen als dat de software betrouwbaarheid ten goed komt. Er is sprake van een systems engineering approach voor de overall ontwikkeling van het systeem. *Projectorganisatie en aansturing daarvan lieten ruimte voor redesign, indien noodzakelijk. Allen betrokken partijen waren capabel en betrokken.*	-½

Product		
7	**Complexiteit beslislogica** 4 Beslislogica is redelijk eenvoudig (bevat enkele zeer geïsoleerde uitzonderingssituaties), McCabe index tussen 10 en 30.	-⅓
8	**Omvang software module (Lines of code)** 3 Tussen 10.000 en 50.000	+⅓
9	**Helderheid gebruikte architectuurconcepten** 4 Er is een scheiding van taakuitvoering tussen modulen beschreven, welke het principe van "maximale cohesie en minimale koppeling respecteert", maar deze is passief bewaakt tijdens het ontwikkelproces	0
10	**Gebruik van een gecertificeerde compiler** 3 Gebruik van een compiler waar de ontwikkelaar langdurige ervaring mee heeft	0

Requirements traceability / verifieerbaarheid	
11 **Traceerbaarheid van requirements door het proces heen** 4 Aantoonbaar traceerbaar naar architectuur en testen	-⅓

Testen	
12 **Testtechnieken en dekkingsgraad** 5 Formele testtechniek(en) gehanteerd met gemiddelde dekkingsgraad	-⅔

Executieomgeving / gebruik	
13 **Multiprocesomgeving** 2 Meerdere applicaties op één stuk hardware	+⅓
14 **Aanwezigheid representatieve velddata gedurende missie** 5 Veel representatieve velddata aanwezig en gebruikt van identieke of sterk vergelijkbare toepassingen	-2
15 **Monitoring** 3 Weinig/kort gemonitord gedurende taakuitvoering	0
Totaal	-4½

Dit geeft aanleiding tot een geschatte faalkans van $10^{-4½} = 3{,}2.10^{-5}$, wat volledig overeenkomt met de inschatting door experts die nauw bij het project betrokken zijn.

6.2 Component “off-the-shelf”

Kijkt men naar een COTS model van een bekende leverancier, denk aan bijvoorbeeld klepbesturingen die op een dedicated PLC wordt geleverd, dan is het nog steeds mogelijk een berekening uit te voeren. Men heeft alleen minder informatie, wat aanleiding geeft tot het wegvallen van veel factoren. Op basis van de wel beschikbare (of geschatte) informatie krijgt men het volgende beeld.

Totstandkomingproces		
4	**Cultuur en samenwerking** 4 Zelflerende organisatie *Gebaseerd op ervaren werkwijze tijdens uitrol product*	-½
5	**Opleidingsniveau en ervaring ontwikkelaars** 5 Aantoonbaar uitstekende kennis en veel ervaring van het specifieke domein *Gebaseerd op reputatie ontwikkelorganisatie (naam in het veld van bedrijf), in combinatie met ervaringen tijdens bouw systeem en uitrol product*	-½
Product		
7	**Complexiteit beslislogica** 5 Beslislogica en foutherkenning zijn erg eenvoudig, McCabe Index kleiner dan 10. *Gebaseerd op een inschatting aan de hand van de te automatiseren beslislogica: de uit te voeren handeling is zeer recht-toe-recht-aan.*	-½
8	**Omvang softwaremodule (Lines of code)** 6 Minder dan 1000 *Gebaseerd op een inschatting aan de hand van een Functiepunten-analyse.*	-½
Executieomgeving / gebruik		
13	**Multiprocesomgeving** 5 Dedicated CPU en memory op geen of een triviaal OS *Gebaseerd op het feit dat de applicatie op een dedicated PLC geleverd wordt.*	-⅓
14	**Aanwezigheid representatieve velddata gedurende taakuitvoering** 4 Veel representatieve velddata aanwezig en gebruikt van identieke of sterk vergelijkbare toepassingen *Gebaseerd op het grootschalige gebruik van de klepbesturing bij veel installaties in de olie-industrie, in vergelijkbare fysieke omgevingen en risicoklasse.*	-2
Totaal		-4⅓

Dit geeft aanleiding tot een geschatte faalkans van $10^{-4⅓}$ = $4{,}6.10^{-5}$, wat volledig overeenkomt met de inschatting door experts.

7 Referenties

[1] S.E. van Manen e.a., *TOPAAS de theorie,* versie 2.0,RWS DI 1-4-2011, digitaal doc: TOPAAS - de theorie v2.doc

[2] S.E. van Manen, H.J. van der Graaf, Evaluatie TOPAAS,, versie 3.0, RWS DI 25-2-2011, digitaal doc: Evaluatie pilots v3.doc

[3] IEEE1471 *Recommended Practice for Architecture Description of Software-Intensive Systems*, IEEE, 2000

[4] P. Kruchten, *Architectural Blueprints—The "4+1" View Model of Software Architecture*, IEEE Software 12 (6), November 1995, pp. 42-50

[5] G. Heslinga, *Faalkansmodel voor menselijke fouten, toegespitst op menselijk handelen bij Haringvliet- en Volkeraksluizencomplex*, Dok. nr. 8140-06-062, versie : 1.1, 7 september 2006

[6] IEC61508 *Functional safety of electrical/electronic/programmable electronic safety-related systems*, CEI/IEC, 1998

[7] NESMA, *Definities en Telrichtlijnen voor de toepassing van functiepuntanalyse. NESMA Functional Size Measurement method conform ISO/IEC 24570*, ISBN: 978-90-76258-17-1, 2003

Bijlage A Het Fagan-inspectieproces

De Fagan-inspectie (ooit door Michael Fagan voor IBM bedacht) is een software review methode waarbij een (tussen)product, bijvoorbeeld een ontwerp of code, grondig wordt bekeken met het doel om fouten te corrigeren voordat het project naar een volgende ontwikkelfase doorgaat. De inspectie wordt na elke ontwikkelfase uitgevoerd door drie tot zes mensen met rollen als moderator, inspecteur, notulist of auteur. Het proces bestaat uit zes stappen:

1. **Planning.** De moderator plant de inspectie. Hij stelt de inspectie team samen, plant de bijeenkomsten en verspreidt het benodigde materiaal.
2. **Aftrap** [optioneel]. Zo nodig wordt het te inspecteren product toegelicht door de auteur.
3. **Voorbereiding**. De inspecteurs bereiden zich individueel voor op de inspectiebijeenkomst. Ze bestuderen het product vanuit de eigen expertise en zoeken naar fouten, met behulp van de 'hogere documenten' en de controlelijsten. Na afloop vullen ze de gevonden fouten en de benodigde tijd in op het individuele bevindingenformulier. De moderator verzamelt alle individuele bevindingenformulieren.
4. **Inspectie**. De inspecteurs zoeken, bespreken en classificeren gezamenlijk de fouten in het (tussen)product. De moderator leidt de bijenkomst en zorgt dat er naar fouten gezocht moet worden - niet naar oplossingen - en dat de auteur niet daarop aangevallen wordt. De notulist noteert de gevonden fouten op een registratieformulier.
5. **Herstel**. De auteur herstelt alle gevonden ernstige fouten en houdt bij welke fouten gecorrigeerd zijn en hoeveel tijd het heeft gekost.
6. **Afronding**. De moderator controleert of de auteur het herstelwerk goed heeft gedaan en of het product aan de eindcriteria voldoet. Verder maakt de moderator een samenvatting van de inspectie waarin onder andere worden vermeld het aantal deelnemers, de bestede tijd per stap, het aantal fouten, de hersteltijd en of het product geaccepteerd is.

Typerend voor Fagan-inspecties is het gebruik van start- en eindcriteria die het begin en het afronden van een ontwikkelfase conditioneren. Die komen voort uit de randvoorwaarden die in de ‘hogere documenten’, namelijk de documentatie van het hogere abstractieniveau, beschreven staan. Een gedetailleerd ontwerp is bijvoorbeeld het 'hogere document' van code.
Uit de praktijk blijkt dat het inspectieresultaat sterk afhangt van de nauwkeurigheid waarmee het proces uitgevoerd wordt. Daarom is het belangrijk dat de mensen die ermee beginnen, getraind zijn in het uitvoeren van Fagan-inspecties en over een minimale infrastructuur beschikken: standaard inspectieformulieren, controlelijsten en een database om de gegevens in op te slaan.
Studies bij verschillende bedrijven (zoals AT&T, HP, NASA, IBM, ICL, BNR) hebben aangetoond dat Fagan-inspecties 5 tot 20 keer effectiever zijn dan testen, de productiviteit verhogen met 14 tot 23 procent en het fouten herstellen 10 keer efficiënter maakt.