

Verslag Tinlab Advanced Algorithms

Galvin Bartes 0799967
176-671

14 december 2023



Inhoudsopgave

1	Inleiding	2
2	Theoretisch kader	3
2.1	Begrippen, tools en literatuur	3
3	Requirements	5
3.1	Requirements engineering	5
3.2	Requirements	5
3.3	specificaties	5
3.3.1	Functional vs non functional requirements	6
3.3.2	Safety critical systems	6
3.4	Rampen	11
3.4.1	Therac-25	11
3.4.2	Ethiopian Airlines Flight 302,boeing 737 crashes	12
3.4.3	China explosie 2015 Tianjin	13
3.4.4	schipholbrand	14
3.4.5	1951	14
3.4.6	slmramp	15
3.4.7	Tsjernobyl	15
4	Modellen	16
4.1	De Kripke structuur	16
4.1.1	CTL	16
4.2	Guards en invarianten	17
4.3	Deadlock	17
4.4	Zeno gedrag	17
5	Logica	17
5.1	Propositielogica	17
5.2	Predicatenlogica	18
5.3	Kwantoren	18
5.4	Dualiteiten	19
5.5	Operator: AG	19
5.6	Operator: EG	20
5.7	Operator: AF	20
5.8	Operator: EF	20
5.9	Operator: AX	20
5.10	Operator: EX	20
5.11	Operator: $p \cup q$	20
5.12	Operator: $p \cap q$	21
5.13	Fairness	21
5.14	Safety	21
5.15	liveness properties	22

1 Inleiding

In dit verslag ga ik in op de kennis en achtergrondinformatie die nodig is voor het toepassen van Time-based model technieken. Door de toenemende complexiteit van systemen is het gebruik van modellen en de toepassing van timebased model checking op industriële controle systemen een manier van modelleren van het systeem en de requirements zodat er een bijdrage kan worden geleverd aan de acceptatie van simulatie-/modeltechniek voor de industrie.[?]. De behandelde onderwerpen zijn requirements-engineering, computational Tree Logic, propositielogica en predicatenlogica.

2 Theoretisch kader

In dit hoofdstuk houdt ik me bezig met de bestudering van rampen aan de hand van het vier-variabelen model maakt het analyseren mogelijk van rampsituaties. Van een aantal rampen is een beschrijving gegeven met datum, plaats en oorzaak. De analyse van de 4-variabelen modellen zal gebruikt worden voor de requirementsdefinitie, ontwerp en ontwikkeling van het sluismodel.

2.1 Begrippen, tools en literatuur

MODE CONFUSION Mode confusion treed op als geobserveerd gedrag van een technisch systeem niet past in het gedragspatroon dat de gebruiker in zijn beeldvorming heeft en ook niet met voorstellingsvermogen kan bevatten.

Wat is automatiseringsparadox Gemak dient de mens. Als er veel energie wordt gestoken in de ontwikkeling van hulpmiddelen die taken van werknemers overemen heeft dat tot resultaat dat veel productieprocessen worden geautomatiseerd. De vraag is dan of vanuit mechnisch wereldpunt de robot niet de rol van de mens overneemt en of de mens nog de kwaliteiten heeft om het werk zelf te doen. [?] [?]

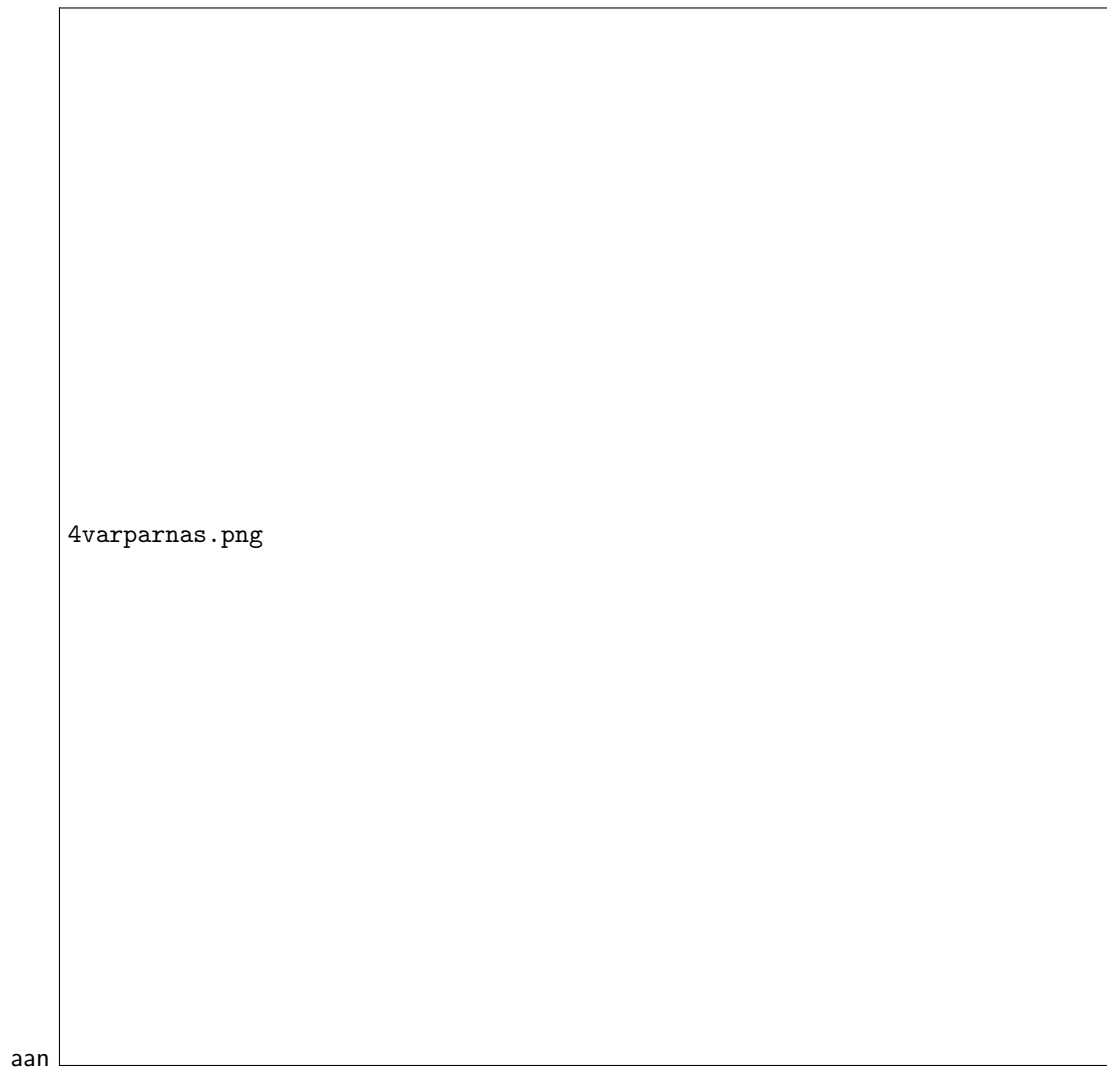
4 variabelen model Er zijn veel veiligheidskritische computersystemen nodig voor het bewaken en besturen van fysieke processen processen. Het viervariabelenmodel, dat al bijna met succes in de industrie wordt gebruikt veertig jaar, helpt het gedrag van en de grenzen tussen het fysieke te verduidelijken processen, invoer-/uitvoerapparaten en software. [?]

Het 4 variabelen model kort toegelicht Monitored variabelen: door sensoren gekwantificeerde fenomenen uit de omgeving, bijv temperatuur

Controlled variabelen: door actuatoren bestuurde fenomenen uit de omgeving Gecontroleerde variabelen kunnen bijvoorbeeld de druk en de temperatuur zijn in een kernreactor, terwijl gecontroleerde variabelen ook visuele en hoorbare alarmen kunnen zijn als het uitschakelsignaal dat een reactorsluiting initieert; wanneer de temperatuur of druk bereikt abnormale waarden, de alarmen gaan af en de uitschakelprocedure wordt gestart

Input variabelen: data die de software als input gebruikt Hier modelleert IN de ingangshardware-interface (sensoren en analoog-naar-digitaal-omzetters) en relateert waarden van bewaakte variabelen aan waarden van invoervariabelen in de software. De invoervariabelen modelleren de informatie over de omgeving die beschikbaar is voor de software. Bijvoorbeeld, IN zou een druksensor kunnen modelleren die temperatuurwaarden omzet in analoge spanningen; Deze spanningen worden vervolgens via een A/D-omzetter omgezet in gehele waarden die zijn opgeslagen in een register dat toegankelijk is voor de computer software.

Output variabelen: data die de software levert als output De uitgangshardware-interface (digitaal-naar-analoog converters en actuatoren) is gemodelleerd door OUT, dat waarden van de uitvoervariabelen van de software relateert aan waarden van gecontroleerde variabelen. Een uitvoervariabele kan bijvoorbeeld een Booleaanse variabele zijn die door de software is ingesteld met de begrippen dat de waarde true aangeeft dat een reactorsluiting zou moeten plaatsvinden en de waarde false geeft het tegenovergestelde



6 Variable model Een uitbreiding van een 4-variabelen model is het 6-variabelen model. Optitatie statements omschrijven de omgeving zoals we het willen zien vanwege de machine. Indicatieve statements omschrijven de omgeving zoals deze is los van de machine. Een requirement is een optitatie statement omdat ten doel heeft om de klantwens uit te drukken in een softwareontwikkel project. Domein kennis bestaat uit indicatieve uitspraken die vanuit het oogpunt van software ontwikkeling relevant zijn. Een specificatie is een optitatie statement met als doel direct implementeerbaar te zijn en ter verondersteuning van het nastreven en realiseren van de requirements. Drie verschillende type domein-kennis: domein eigenschappen, domein hypothesen, en verwachtingen. Domein eigenschappen zijn beschrijvende statementsover een omgeving en zijn feiten. Domein hypothesen zijn ook beschrijvende uitspraken over een omgeving, maar zijn aannames. Verwachtingen zijn ook aannames, maar dat zijn voorschrijvende uitspraken die behaald worden door actoren als personen, sensoren en actuators.

3 Requirements

3.1 Requirements engineering

Requirementsengineering gaat over alle activiteiten gericht op het identificeren van user requirements, analyseren van de requirements ,documentatie van requirements en de specificaties hiervan, validering van de gedocumenteerde requirements tegenover gebruikersbehoefte en ook processen die deze activiteiten ondersteunen. Requirementsengineering kan gebruikt worden voor software, hardware en elektromechanische systemen. requirements engineering houdt zich bezig met alle fasen van project- of product-ontwikkeling levenscycli van innovatie tot veroudering.

Door middel van requirements wordt vastgelegd welke functionaliteiten het systeem moet bevatten volgens de stakeholders, welke prestatieniveaus het moet bereiken, welke beperkingen moeten worden gehandhaafd en welke kwaliteitsattributen moeten worden nageleefd. Stakeholders zijn onder andere managers, eindgebruikers, functioneel beheerders, technisch beheerders, operationeel beheerders, helpdesk, ontwikkelaars en audit medewerkers. Het formuleren en vaststellen van requirements is een cruciale fase in het systeemmodelleringsproces, omdat het een duidelijk beeld geeft van de gewenste systeemeigenschappen en helpt bij het bepalen van de benodigde functionaliteit, prestaties alsook de keuze voor de ontwikkelomgeving. Dit legt de basis voor verdere stappen in het modelleringsproces, zoals ontwerp, implementatie en testen.

3.2 Requirements

In het modelleren van systemen verwijst een requirement naar een specifieke functionaliteit, prestatie-eis, beperking of kwaliteitsattribuut dat aanwezig moet zijn in het systeem [?]. In dit verslag maken we onderscheid tussen de requirements en specificaties. Een requirements is een formele beschrijving van wat het systeem moet kunnen doen en een specificatie gaat over hoe het systeem werkt en met welke kwaliteitseigenschappen. Een requirement wordt vastgelegd in een specificatiedocument. Het opstellen van specificaties[?] is een belangrijk proces, omdat het de basis legt voor het ontwerp en de ontwikkeling van het systeem. Door duidelijke specificaties te hebben, kunnen de betrokken stakeholders een gemeenschappelijk begrip hebben van wat er van het systeem wordt verwacht en in hoeverre dit veilig en haalbaar is.

3.3 specificaties

Specificaties zijn gedetailleerde beschrijvingen van wat het systeem moet kunnen doen, welke prestaties het moet leveren en welke beperkingen er gelden. Ze dienen als een leidraad tijdens het ontwikkelingsproces en zorgen voor een duidelijke communicatie tussen de belanghebbenden over de gewenste resultaten.

Door middel van specificaties wordt vastgelegd welke functionaliteiten het systeem moet bevatten, welke prestatieniveaus het moet bereiken, welke beperkingen moeten worden gehandhaafd en welke kwaliteitsattributen moeten worden nageleefd.

Het is ook belangrijk om de haalbaarheid van de specificaties te overwegen. Sommige eisen kunnen technisch uitdagend zijn of beperkt worden door bijvoorbeeld budgettaire of tijdsbeperkingen. Daarom moeten de specificaties realistisch en praktisch uitvoerbaar zijn. Voor een systeem moeten de requirements gedocumenteerd zijn, traceerbaar zijn, gereviewed, en getest.

3.3.1 Functional vs non functional requirements

Functionele requirements gaan over de functie die een systeem moet uitvoeren, terwijl non-functionele requirements betrekking hebben op de algemene eigenschappen van een systeem, zoals snelheid, bruikbaarheid, veiligheid, betrouwbaarheid. Non-functionele eisen worden ook wel systeemkwaliteiten betekend.

3.3.2 Safety critical systems

Een veiligheidsanalyse is een manier voor het evalueren van ongelukken en risico's op verschillende niveaus. Met een veiligheidsanalyse wordt gekeken naar mensen en systemen die worden blootgesteld aan een gevaar/risico en de mogelijkheid deze te minimaliseren. Veiligheid is moeilijk te definiëren omdat het een vaag concept is. De kennis van de mens is verantwoordelijk voor het scheppen van veiligheid. Kennisdoelen zijn daarom altijd belangrijk en kunnen worden onderscheiden in 3 domeinen: cognitief, affectief en psychomotorisch.[?] Een veiligheidsanalist moet kennis hebben van het systeem, maar ook ervaring met het systeem en vooral hoe een fout in het systeem zich voordoet. Volgens Stoney[?] is veiligheid een aspect dat de veiligheid van mensenlevens of de omgeving niet in gevaar brengt. De interactie tussen systeem en omgeving levert kennis op door ad-hoc ervaring. Maar ook is kennis van abstractie en systeemdoelen belangrijk. Het kwalificeren van informatie moet ook op waarde geschat worden. Een verkeerde schatting kan fataal zijn. Zelfs als een veiligheidsanalyse is gedaan waarbij risico's zijn gedefinieerd en geprioritiseerd is een definitie van een veiligheidsniveau nodig. Soms zijn hier richtlijnen voor en soms ook niet. Er zijn voorbeelden van tools en technieken te gebruiken voor een veiligheidsanalyse. En dat zijn

1. checklists voor kritische veiligheidselementen
2. fault-boom analyse
3. event-boom analyse
4. fouttoestand en kritische effect-analyse (FMECA, FMEA)

[?]

Schermafbeelding 2023-12-08 144723.png

Schermafbeelding 2023-12-08 133303.png

Schermafbeelding 2023-12-08 132959.png

Schermafbeelding 2023-12-08 132840.png



Schermafbeelding 2023-12-08 144723.png

[?]
[?] [?] [?] [?] [?] [?].

3.4 Rampen

Voor deze studie is onderzoek gedaan naar verschillende rampen aan de hand van het vier variabelen model. Elke ramp op deze manier categoriseren kan ons helpen te bepalen in hoeverre requirements een rol kunnen spelen in de veiligheid van ons model.

3.4.1 Therac-25

Beschrijving In de periode van Juni 1985 and Januari 1987 zijn er meerdere ongelukken met dodelijke afloop door de implementatie van de Therac-25 bij de behandeling van huidkanker. Dit apparaat gebruikt

elektronen om stralen met hoge energie te creëren die tumoren kunnen vernietigen met minimale impact op het omliggende gezonde weefsel.

Datum en Plaats De ongelukken vonden plaats in de periode van juni 1985 tot en met januari 1987.

Oorzaak Onderzoekers constateren dat er fouten zijn gemaakt tijdens de (her-)implementatie van systemen uit eerdere productiemodellen. Terwijl de therac 20 afhankelijk was van mechanische vergrendelingen werd er bij de therac-25 software gebruikt. Onderzoekers komen daarom tot de volgende conclusies Software problemen zijn onder andere:

- slechte software engineering/designing praktijken
- de machine is afhankelijk van software voor veiligheidsoperaties
- de fout in de code is niet zo belangrijk als een geheel onveilig ontwerp
- het reinigen van de buigmagneetvariabele in plaats van aan het uiteinde van het frame
- raceconditionering om aan te geven dat het invoeren van het recept nog steeds aan de gang is
- reactie van de gebruiker
- slechte subroutines voor schermvernieuwing die foutieve informatie op de werkende console achterlieten
- Problemen met het laden van tapes bij het opstarten, waarbij het gebruik van photom-tabellen werd uitgesloten om het interlock-systeem te activeren in het geval van een laadfout in plaats van een checksum

Onderzoekers zijn van mening dat de tekortkomingen in medische apparatuur niet geheel en altijd te verwijten zijn aan softwareproblemen. Zo is er ook een rol weggelegd voor fabrikanten en overheden. Klankbordgroepen klagen over het tekort aan software-evaluaties en een tekort aan hard-copy audit trials om foutmeldingen in beeld te krijgen. [?]

3.4.2 Ethiopian Airlines Flight 302, Boeing 737 crashes

Beschrijving Ethiopische Airlines-vlucht 302 was een geplande internationale passagiersvlucht van Bole International Airport in Addis Abeba, Ethiopië naar Jomo Kenyatta International Airport in Nairobi, Kenia. Op 10 maart 2019 stortte het Boeing 737 MAX 8-vliegtuig dat de vlucht uitvoerde zes minuten na het opstijgen neer nabij de stad Bishoftu, waarbij alle 157 mensen aan boord omkwamen.

Vlucht 302 is het dodelijkste ongeval van Ethiopië Airlines tot nu toe en overtreft de fatale kaping van vlucht 961, resulterend in een crash nabij de Comoren in 1996. Het is ook het dodelijkste vliegtuigongeluk dat zich in Ethiopië heeft voorgedaan, en overtreft de crash van een Antonov An-26 van de Ethiopische luchtmacht in 1982, waarbij 73 mensen omkwamen.

Dit was het tweede MAX 8-ongeluk in minder dan vijf maanden na de crash van Lion Air-vlucht 610. Beide crashes waren aanleiding voor een twee jaar durende wereldwijde langdurige aan de grond houden van het vliegtuig en een onderzoek naar de manier waarop het vliegtuig werd goedgekeurd voor passagiersvervoer.

Datum en Plaats De ramp vond plaats op 10 maart 2019 nabij de stad Bishoftu

Oorzaak De technische oorzaak ligt bij het MCAS flight control system. Dit systeem werd geïmplementeerd om kosten te reduceren en opleidingen voor piloten in te korten. Deze single point of failure [?] werd getriggerd door een enkele angle-of-attack sensor[?]. Bij de nieuwe boeing 737 max model werden tests uitgevoerd in volledige flight simulators. Nieuwe regels van het FAA instituut vereisten ondersteuning bij het uitvoeren van enkele manoeuvres. Tijdens testvluchten uitgevoerd binnen een jaar voor certificatie werd het pitch-up fenomeen geconstateerd waarop het mcas systeem werd aangepast. In het systeem zaten nu de volgende fouten.

- MCAS wordt getriggerd voor enkele sensor zonder vertraging
- Het ontwerp staat toe dat in situaties waar de angle-of-attack fout is de mCAS wordt geactiveerd
- systeem kreeg onnodig bevoegdheid controle om de neus bij te sturen
- Waarschuwingsscherm bij fouten in de angle-of-attack werkte niet door softwarefout. Het werd ook niet kritisch bevonden door ethiopian airlines. geplande updates door boeing pas in 2020
- een losstaande fout in de microprocessor van de controle computer kan vergelijkbare situaties doen voorkomen zonder dat mcas wordt geactiveerd

Fouten vielen niet op omdat FAA test uitbesteedde aan Boeing. Contact tussen de organisaties FAA en Boeing verliep op management niveau. Boeing instrueerde niet alle piloten over MCAS. En het MCAS systeem werd gezien als een achtergrond systeem.

Uit onderzoek is op te maken dat problemen bij boeing toestellen te verwijten zijn aan:

- marktdruk
- slapshot engineering
- missiekritische toepassing van technologie
- geen regelgevingsregime voor foutieve risicoanalyse

3.4.3 China explosie 2015 Tianjin

Beschrijving De explosie zorgde voor de vernietiging van 12000 voertuigen, schade aan 17000 huizen binnen een straal van 1 km. Er waren 173 doden inclusief brandweermensen. Een van de explosies zorgde voor een beving van 2.3 op de schaal van richter. Opgeslagen materialen waren: calcium carbide, sodium nitraat, potassium nitraat, ammoniak nitraat en cyanide. Ook is er veel kritiek geweest op de acties van de autoriteiten. Zo was er censuur vanuit de overheid op de journalistiek. Ook was er naar alle waarschijnlijkheid sprake van corruptie. Zo bleek achteraf dat een van de grootste aandeelhouders Dong Shexuang de zoon is van een oud-politiekop in Tanjin haven, genaamd Dong Pijun

Datum en Plaats De ramp vltrok zich pp 12 augustus 2015 bij de Rulthai logistiek. Deze faciliteit zorgde voor de opslag van gevaarlijke stoffen.

Oorzaak De volgende factoren zouden een rol hebben gespeeld:

- Een onjuiste afbakening van het opslagmateriaal
- Er was weinig kennis bij de autoriteiten over opslagmaterialen. Zo bleek er 7000 ton aan materiaal opgeslagen, dat is ruim 70 keer te maximaal toegestane hoeveelheid.
- Onverenigbaar grondgebruik in de nabije omgeving. Veel woonwijken met naar schatting 6000000 bewoners en 500 lokale bedrijven in de buurt van de opslag gevaarlijke stoffen.

3.4.4 schipholbrand

Beschrijving Bij de schipholbrand op 27/10/2005 vielen zeker 11 doden onder migranten in de cellencomplexen van schiphol-oost. Doodsoorzaak van de slachtoffers is verstikking. Het gebouw voldeed niet aan de eisen voor brandveiligheid, personeel was niet goed getraind voor dergelijke situaties en de hulpverlening kwam door verschillende factoren te laat op gang.

Datum en Plaats De ramp voltrok zich bij de vleugels j en k van het cellencomplexen van schiphol-oost op 27 oktober 2005 .

Oorzaak Doordat de deur van cel 11 niet werd gesloten kreeg de brand zuurstof. De brand kon zich uitbreiden in de aanwezigheid van grote hoeveelheid brandbare materialen. Deze fungeerde als brandstof waardoor de brand zich verder kon ontwikkelen zowel binnen als buiten de cel. De dakluiken werkten met als gevolg dat de rook en warmte niet werden afgevoerd. Dit heeft het mede onmogelijk gemaakt alle celdeuren te openen en daardoor de reddingsoperatie van de bewaarders belemmerd. Door de schilconstructie van het cellencomplex was het mogelijk dat de brand zich kon uitbreiden naar de andere cellen en naar de gang.

Dit is onopgemerkt gebleven doordat er geen specifieke eisen werden gesteld bij de bouw van vleugels j en k. Was er te weinig aandacht voor bouwregels en onderzoek naar eerdere branden in het cellencomplex. Bovendien waren bhw'ers niet goed opgeleid en was er geen goede afstemming met de brandweer. Er was geen evacuatieplan dat dat rekening hield met de overplaatsing van gedetineerden naar ander penitentiare inrichtingen. En had men te weinig aandacht voor de traceerbaarheid van celbewoners en bovendien voor ademhalingsproblemen van gedetineerden.

3.4.5 1951

Beschrijving Turkish Airlines-vlucht 1951 (ook bekend als TK 1951) was een passagiersvlucht die op woensdag 25 februari 2009 neerstortte in de buurt van het Nederlandse vliegveld Schiphol. Hierbij kwamen 9 van de 135 inzittenden om het leven.

Het vliegtuig, een Boeing 737-800 van de Turkse maatschappij Turkish Airlines, was op weg van het vliegveld Istanbul Atatürk naar Schiphol, maar stortte om 10:26 uur, kort voor de landing op de Polderbaan, neer op een akker en brak daarbij in drie stukken.

Datum en Plaats Op woensdag 25 februari 2009 voltrok zich de ramp met een toestel van turkisch airlines tijdens vlucht 1951.

Oorzaak De automatische reactie van het toestel werd getriggerd door een fout gevoelige radio altimeter waardoor de automatische gashendel de energiemotor op actief stelde. Inadequaats handelen van de piloten ondanks een defecte hoogtemeter en onvolledige instructies van de luchtverkeersleiding.

Het officiële onderzoek van de Onderzoeksraad voor Veiligheid wees inadequaats handelen van de piloten als hoofdoorzaak aan voor het ongeluk. Ondanks een defecte hoogtemeter en onvolledige instructies van de luchtverkeersleiding hadden de piloten het ongeluk kunnen voorkomen, aldus de Onderzoeksraad.

3.4.6 slm ramp

Beschrijving Toen het toestel op 07/06/1989 de Anthony Nesty Zanderij naderde, was het daar, anders dan het weerbericht had voorspeld, mistig. Het zicht was evenwel niet zo slecht dat er niet op zicht kon worden geland. Gezagvoerder Will Rogers besloot echter via het Instrument Landing System (ILS) te landen, hoewel dit niet betrouwbaar was en hij voor zo'n landing ook geen toestemming had. De gezagvoerder brak drie landingspogingen af. Bij de vierde poging negeerde de bemanning de automatische waarschuwing (GPWS) dat het toestel te laag vloog. Het toestel raakte op 25 meter hoogte twee bomen. Het rolde om de lengte-as en stortte om 04.27 uur plaatselijke tijd ondersteboven neer.

Datum en Plaats De ramp voltrok zich op 07 juni 1989 toen het vliegtuig de Anthony Nesty Zanderij naderde

Oorzaak

Uit onderzoek bleek dat de papieren van de bemanning niet in orde waren door nalatigheid in de crew-member screening bij de SLM. Geconcludeerd werd dat de gezagvoerder roekeloos had gehandeld door voor een ILS-landing te kiezen terwijl hij daar geen toestemming voor had, en door onvoldoende op de vlieghoogte te hebben gelet. Het vliegen onder de minimum hoogte leidde tot collision met een boom.

3.4.7 Tsjernobyl

Beschrijving De mislukte veiligheidscontrole die 26 april 1986 01.24 uur in de Sovjet-Unie leidde tot explosies in een van de reactoren in de kerncentrale. De reactoren hadden geen veiligheidsomhulling en de reactor bevat grote hoeveelheden brandbaar grafiet. Door de explosie en de brand kwamen er radioactieve stoffen vrij. Het gaat helemaal mis in de kernreactor 4. De warmteproductie nam toe met een explosie tot gevolg. 31 mensen kwamen om, waaronder veel mensen dagen later door stralingsziekte.

Datum en Plaats De ramp van Tsjernobyl voltrok zich op 26 april 1986 [?].

Oorzaak

Technici bij kerncentrale 4 voerden een slecht opgezet/ ontworpen experiment uit. De kracht regulering werd uitgeschakeld evenals veiligheidssystemen. Door een Bedieningsfout in een testprocedure werd het vermogen van de koelinstallaties negatief beïnvloed. Mede door een ontwerpfout in de noodstopprocedure kon in het systeem niet snel genoeg schakelen om remmende invloed uit te oefenen op het toenemende vermogen van de reactorkernen. Met brand en explosies tot gevolg. [?] [?] [?] [?] [?]

4 Modellen

4.1 De Kripke structuur

De kripke structuur is een set van locaties, transities, guards, klokken en data-variabelen .

We gebruiken een Kripke-structuur, om het gedrag van reactieve systemen vast te leggen. Een kripke-structuur bestaat uit een reeks toestanden, een reeks overgangen tussen toestanden, en een functie die elke toestand een reeks eigenschappen geeft die 'geldig' zijn in deze toestand. Paden in een kripke-structuur modelleren staan voor de transities van het systeem.

4.1.1 CTL

Computational Tree Logic(CTL), geïntroduceerd door Emerson en Clarke , is een logica die wordt gebruikt als specificatietaal voor model checking. CTL-formules zijn samengesteld uit logische operatoren, padkwantificatoren en tijdelijke operatoren. De kwantificatoren worden gebruikt om aan te geven of een eigenschap van het systeem geldig is op een bepaald rekenpad (E) of alle rekenpaden (A) van een locatie. Enkele temporele operatoren gebruikt om eigenschappen te beschrijven van een pad zijn AX, EX, AU, EU, AG, EG, AF, en EF. Met deze kwantoren is mogelijk specificaties van een requirement te vertalen naar een formele specificatie. Op deze manier is een systeem van een model testbaar op correctheid, reliability, safety, deadlock, responsiveness en liveness requirements.

Model en verificatieomgeving Een model kan een weergave zijn voor een computersysteem uit het dagelijks leven. Dit kunnen systemen met kritische functies zijn zoals auto's vliegtuigen en industriële installaties, maar ook andere systemen waarbij timingaspecten van cruciaal belang zijn kunnen gemodelleerd worden zoals telecommunicatie, security protocollen, medical equipment, factory automation, transportation, communications, automotive, e-commerce applicaties en privacy. Formele verificatie is een systematisch proces waarin bevestigd wordt of het bestudeerde systeem voldoet aan de specificaties of requirements vastgesteld in het specificatiedocument. Een formele specificatie is een complete representatie van een systeem waarbij bepaalde formele specificaties worden vertaald. Verschillende specificatie technieken die gebruikt worden als formele methode zijn abstracte state machines zoals abstract state machine models, automata based models, object oriented models etc. Daarnaast zijn er ook op automata gebaseerde modellen welke gebruik maken van deterministic finite automata, nondeterministic finite automata, timed automata, Buchi automata, -automata etc. In model checking wordt een gedragseigenschap geverifieerd over een model door impliciet of expliciet alle bereikbare locaties na te lopen die naar een dergelijke state/locatie kunnen leiden. De eigenschappen worden uitgedrukt in formules voor temporele logica. Deze formele verificatie aan de hand van temporele logica, is een systeemproces dat gebruik maakt van wiskundig redenerenverifiëren of een systeem (dat wil zeggen het model) aan een bepaalde vereiste, de getimede temporele eigenschappen, voldoet. De model checker bevestigt dat de eigenschappen geldig zijn of geeft aan dat deze niet geldig zijn.

In de formele verificatieomgeving wordt het mogelijk gemaakt om te controleren of het op tijd-automaten gebaseerde model met de temporele logische eigenschappen voldoet aan de gedefinieerde eigenschappen.

Een goed voorbeeld van een verificatieomgeving is Uppaal. Uppaal is een tool voor het modelleren, simuleren en verifiëren van real-time controllers en communicatieprotocollen, , gezamenlijk ontwikkeld door Basic Research in Computer Science aan de Universiteit van Aalborg in Denemarken en de afdeling Informatietechnologie aan de Universiteit van Uppsala in Zweden.

Behalve clocks, ondersteund de tool zowel simple and complex data types zoals bounded integers, arrays, synchronization via shared variabelen en actions schedulability.

4.2 Guards en invarianten

Invarianten en guards zijn condities. Een guard is een conditie die geldt in een transitie. De transitie kan alleen genomen worden wanneer de guard geldig is. Een invariant is een conditie die geldt in een state en die altijd waar is wanneer de automaat zich in die state bevindt.

Invarianten zijn eigenschappen die worden gegeven door een voorwaarde voor de toestanden en vereisten geldend voor alle bereikbare staten. Het begrip 'invariant' kan dus als volgt worden uitgelegd: er moet aan de voorwaarde ϕ zijn voldaan door alle begintoestanden en de bevrediging van ϕ is onveranderlijk onder alle overgangen in het bereikbare pad van het gegeven transitiesysteem. Invarianten kunnen worden gezien als toestandseigenschappen en kan worden gecontroleerd door te kijken naar de bereikbare staten.

4.3 Deadlock

Een deadlock is een bereikbare staat waarin het systeem helemaal geen acties kan uitvoeren. Een deadlock hangt af van de reeks acties die een bereikbare staat niet kan uitvoeren. Om de impasse te behouden moet A niet alleen overschatten wat P kan doen, maar ook wat P weigert.

4.4 Zeno gedrag

Zeno gedrag houdt in dat de mogelijkheid dat in een eindige hoeveelheid tijd een oneindig aantal handelingen kan worden verricht.

5 Logica

5.1 Propositielogica

In de propositielogica onderzoekt men het waarheidsgehalte van samengestelde uitspraken.

Hierbij worden proposities gemaakt van elementaire proposities en logische voegwoorden. In de symbolische propositielogica, wordt een propositieformule gemaakt van variabelen (letters), logische operatoren en haakjes. Net zoals een formule in de rekenkunde, heeft het deel van een propositieformule binnen de haakjes een hogere prioriteit dan het deel buiten de haakjes. Bovendien wordt alles wat tussen haakjes staat, beschouwd als een eenheid. Om een propositieformule correct samen te stellen, moeten wij verplicht gebruik maken van de volgende grammaticale regels:

1. Een variabele is een formule;
2. Als p een formule is, dan is $\neg p$ een formule;
3. Als p en q formules zijn, dan zijn $(p \wedge q)$, $(p \vee q)$, $(p \rightarrow q)$ en $(p \leftrightarrow q)$ formules.

In een propositieformule mogen haakjes worden weggelaten mits er geen verwarring ontstaat

Tijdens het logisch beschrijven van logica of bij het maken van formules over formules, is het mogelijk een paradoxale uitspraak te doen. Zo'n uitspraak is vergelijkbaar met de volgende zin: "Deze zin is onwaar". De zojuist genoemde zin beschrijft zichzelf. De vraag of deze zin nu waar of onwaar is, is

een paradox. Wij moeten de uitspraken over de logica scheiden van de logica zelf. Daarom worden metasymbolen geïntroduceerd.

Voor het bepalen van de waarheidswaarde van een formule $f(p_1; p_2; \dots; p_n)$ moeten alle mogelijke combinaties van waardetoekenningen worden bepaald. Deze combinatie van waardetoekenningen vormen samen de waarheidstabel van deze formule.

5.2 Predicatenlogica

Hoewel de volgende redenering in de propositielogica ongeldig is, wordt zij intuïtief toch als geldig beschouwd:

1. f_1 "alle republieken plegen geen overspel"
2. f_2 "sommige overspeligen zijn president"
3. y "sommige presidenten zijn geen republiek"

De geldigheid van deze redenering is gebaseerd op informatie, waarmee de propositielogica geen rekening mee houdt. Om deze extra informatie bij het redeneren te betrekken, moeten wij de propositielogica uitbreiden met de begrippen eigenschappen, variabelen en kwantoren. De zin "alle mensen zijn sterfelijk" geeft aan dat objecten, die de eigenschap 'menselijk zijn' hebben, blijkbaar ook de eigenschap 'sterfelijk zijn' hebben. uitspraak symbolisch:

1. sterfelijk objecten $S(x)$
2. menselijke objecten $M(x)$
3. x is een priemgetal $P(x)$

In de uitspraken geven wij de eigenschappen sterfelijk en priemgetal aan met de hoofdletters S en P . De variabele x stelt objecten voor. Een variabele, waarvan de waarde onbekend is, noemen wij vrije variabele. Definitie 4.1 (Predikaat) Een predikaat is een uitspraak met vrije variabelen, die een propositie wordt zodra alle vrije variabelen in dat predikaat gebonden zijn aan een waarde.

5.3 Kwantoren

Kwantificator wordt gebruikt om de variabele van predikaten te kwantificeren. Het bevat een formule, een soort verklaring waarvan de waarheidswaarde kan afhangen van de waarden van sommige variabelen. Wanneer we een vaste waarde aan een predikaat toekennen, wordt het een propositie. Op een andere manier kunnen we zeggen dat als we het predikaat kwantificeren, het predikaat een propositie wordt. Kwantificeren is dus een woordsoort dat verwijst naar kwantificeringen als 'alles' of 'sommige'.

Er zijn hoofdzakelijk twee soorten kwantoren: universele kwantoren en existentiële kwantoren. Daarnaast hebben we ook andere soorten kwantoren, zoals geneste kwantoren en kwantoren in standaard Engels gebruik. Kwantificator wordt voornamelijk gebruikt om aan te tonen dat voor hoeveel elementen een beschreven predikaat waar is. Het laat ook zien dat voor alle mogelijke waarden of voor sommige waarde(n) in het universum van het discours het predikaat waar is of niet.

Tot dusverre hebben wij in de voorbeelduitspraken "voor alle x " en "er is een x " gekoppeld aan alle voorkomens van x in een universum. De uitspraak "er is een x in het universum van x met de eigenschap P " wordt weergegeven als:

Soms willen wij zo'n universum beperken. Zo'n beperkt deel van een universum, een domein, is het gebied waaruit de gebonden variabele moet putten. De beschrijving van het domein $D(x)$ wordt onder de kwantor geplaatst.

Met de universele kwantor \forall bedoelen wij alle waarden van x binnen het opgegeven domein. Als het domein niet wordt opgegeven, dan bedoelen wij een of meer waarden x uit een eindig universum.

Met de existentiële kwantor \exists bedoelen wij 'een' of meer waarden van x uit het opgegeven domein. Als het domein niet wordt opgegeven, dan bedoelen wij een of meer waarden x uit een eindig universum.

Een overzicht van meerdere temporal operators:

1. F sometime in the Future, De operator (F) wordt gebruikt om te specificeren dat een eigenschap ooit geldig zal zijn in een state/locatie van het huidige pad.
2. G Globally in the future, De globale operator (G) wordt gebruikt om te specificeren dat een eigenschap geldig is voor alle states/locaties op het huidige pad.
3. X neXtime, De next time operator (X) wordt gebruikt om te specificeren dat een eigenschap geldig is in de volgende state/locatie.
4. U until, De Until operator (U) wordt gebruikt om te specificeren dat de eerste eigenschap geldig is in alle states/locaties voorafgaand de state waarin de tweede eigenschap geldig is.
5. R the release operator, De release operator "R" wordt gebruikt om te specificeren dat de tweede eigenschap geldig is in alle states/locaties op een pad tot en met de eerstvolgende state waarin de eerste eigenschap geldig is.

Pad quantifiers:

1. E er is een pad waarop een propositie geldig is
2. A voor alle paden is een propositie geldig

5.4 Dualiteiten

Het dual van elke uitspraak in een Booleaanse algebra B is de uitspraak die wordt verkregen door de bewerkingen uit te wisselen $+$ en $,$ en hun identiteitselementen 0 en 1 in de oorspronkelijke verklaring te verwisselen. De dubbele van bijvoorbeeld $(1 + a) (b + 0) = b$ is $(0 \ a) + (b \ 1) = b$. Observeer de symmetrie in de axioma's van een Booleaanse algebra B . Dat wil zeggen, de duale van de reeks axioma's van B is de hetzelfde als de originele set axioma's. Dienovereenkomstig geldt het belangrijke principe van dualiteit in B . Namelijk: Stelling 15.1 (Dualiteitsprincipe): De dualiteit van elke stelling in een Booleaanse algebra is ook een stelling. Met andere woorden: als een uitspraak een gevolg is van de axioma's van een Booleaanse algebra, dan is de duale ook een gevolg van deze axioma's, aangezien de dubbele verklaring kan worden bewezen door de duale van elke stap van het bewijs te gebruiken van de oorspronkelijke verklaring.

5.5 Operator: AG

AG P: Always Globally in alle mogelijke paden, is P altijd waar

5.6 Operator: EG

EGp - in tenminste een pad geldt p is overal geldig. Een pad is een reeks toestanden. Bij elke toestand kan een toestandsformule worden geëvalueerd. Een vertakkende tijdpadformule strekt zich uit over een enkele reeks van opeenvolgende toestanden. In de huidige reeks heeft elke toestand een label p, en is de atomaire toestandsformule p waar in die toestand behorend tot deze reeks.

5.7 Operator: AF

AF p - voor alle paden geldt, p is ergens in de toekomst/ ooit geldig.

Een pad is een reeks toestanden. Bij elke toestand kan een toestandsformule worden geëvalueerd. Deze vertakkende tijdpadformule strekt zich uit over meerdere paden met in de toekomst een geldige opvolgende toestand. Als voor alle paden geldt dat deze vanaf de huidige toestand er in de toekomst een toestand een label p heeft, dan is de atomaire toestandsformule p waar in die toestand.

5.8 Operator: EF

EF, er is een pad waar uiteindelijk, vanaf de huidige staat met enkele andere staten p geldig is EF P: in tenminste 1 pas, zal P vroeg of laat geldig zijn

We hebben hier een pad met een reeks toestanden. Waarbij ooit op dit pad een toestand kan toestandsformule worden geëvalueerd welke een label p heeft, en de atomaire toestandsformule p waar is in die toestand.

5.9 Operator: AX

AXp - voor alle paden is p in de volgende state geldig

We spreken hier van alle mogelijke paden waarvoor geldt dat in de volgende toestand kan een toestandsformule worden als "waar" wordt geëvalueerd. States hebben labels. Als de volgende toestand een label p heeft, dan is de atomaire toestandsformule p waar in die toestand.

5.10 Operator: EX

EX P: in ten minste 1 execution path, is p geldig in de volgende state Een pad is een reeks toestanden. Bij elke toestand kan een toestandsformule worden geëvalueerd. Een vertakkende tijdpadformule strekt zich uit over een mogelijke reeks van opeenvolgende toestanden. Waarbij geldt dat in de huidige reeks er een toestand is met een opvolgende state met een label p heeft, en is de atomaire toestandsformule p waar in die toestand.

5.11 Operator: p U q

A[P U Q]: in alle uitvoerpaden is P geldig totdat Q geldig is. Of E[P U Q]: in tenminste 1 pad is P geldig totdat Q geldig is. Een pad is een reeks toestanden. Bij elke toestand kan een toestandsformule worden geëvalueerd. Een vertakkende tijdpadformule strekt zich in geval van A[P U Q]: uit over alle reeksen van opeenvolgende toestanden. Waarvoor geldt dat als de toestand een label p heeft, dan is de atomaire toestandsformule p waar in die toestand tot aan de toestand waarin een toestand het label q heeft.

5.12 Operator: $p \text{ R } q$

R (“release”) is de logische duaal van de U-operator. De operator vereist dat het tweede argument stand moet houden tot en met de eerste staat waar het eerste argument geldt. Het eerste argument is dat niet nodig om uiteindelijk waar te worden.

Een pad is een reeks toestanden. Bij elke toestand kan een toestandsformule worden geëvalueerd. Een vertakkende tijdpadformule strekt zich uit over een mogelijke reeks van opeenvolgende toestanden. States hebben labels. Als de huidige toestand een label q heeft, dan is de atomaire toestandsformule q waar in die toestand tot aan de toestand waarin p geldig is. Met de voorwaarde dat de eigenschap p niet gelabeld hoeft te worden aan een opvolgende state. De release-operator (R) is ook een binaire operator en betekent dat de tweede argument geldig moet zijn in alle staten tot aan q inclusief de eerste staat die voldoet aan de eerste eigenschap.

5.13 Fairness

Deze eigenschappen beschrijven de vereiste dat een proces vooruitgang boekt in de richting van een specifiek doel, waarvan de verwezenlijking afhangt van de fairness van het systeem. Als een proces nooit wordt uitgevoerd, kan het zijn doel meestal niet bereiken. Daarom worden deze eigenschappen vaak alleen langs eerlijke paden geëvalueerd, d.w.z. paden die voldoen aan een van de volgende drie vereisten voor fairnessconstraints: Absolute Fairness, Impartiality: every process should be executed infinitely often: Strong Fairness: every process that is infinitely often enabled should be executed infinitely often in a state where it is enabled: Weak Fairness: every process that is almost always enabled should be executed infinitely often:

Recall that fairness assumptions (see Section 3.5) are used to rule out certain computations that are considered to be unrealistic for the system under consideration. These unreasonable computations that ignore certain transition alternatives forever are the “unfair” ones; the remaining computations are thus the “fair” ones. As there are different notions of fairness, various distinct forms of fairness can be imposed: unconditional, strong, and weak fairness constraints.

5.14 Safety

Safety properties are often characterized as “nothing bad should happen”. The mutual exclusion property—always at most one process is in its critical section—is a typical safety property. It states that the bad thing (having two or more processes in their critical section simultaneously) never occurs. Another typical safety property is deadlock freedom. For the dining philosophers (see Example 3.2, page 90), for example, such deadlock could be characterized as the situation in which all philosophers are waiting to pick up the second chopstick. This bad (i.e., unwanted) situation should never occur.

Veiligheidseigenschappen worden vaak gekarakteriseerd als ‘er mag niets ergs gebeuren’. Het wederzijdse De uitsluitingseigenschap – er bevindt zich altijd maximaal één proces in de kritieke sectie – is een typische veiligheid eigendom. Er wordt gesteld dat het slechte ding (het hebben van twee of meer processen in hun kritieke sectie tegelijkertijd) komt nooit voor. Een andere typische veiligheidseigenschap is de vrijheid van impasse. Voor de eetfilosofen (zie Voorbeeld 3.2, pagina 90) zou bijvoorbeeld zo'n impasse kunnen zijn gekarakteriseerd als de situatie waarin alle filosofen wachten om de tweede op te pikken eetstokje. Deze slechte (d.w.z. ongewenste) situatie mag nooit voorkomen.

the system cannot reach states that are forbidden by the requirements. Following L. Lamport, a safety property states that something bad must never happen. The “bad thing” represents a critical system state that should never occur, for instance a train being inside a crossing with the gates open.


Liveness properties. Safety properties state what may or may not occur, but do not require that anything ever does happen. Liveness properties state what must occur. The simplest form of a liveness property guarantees that

5.15 liveness properties

Veiligheidseigenschappen geven aan wat er wel of niet mag voorkomen, maar eis niet dat er ooit iets gebeurt. liveness geeft aan wat er moet gebeuren. De eenvoudigste vorm van een liveness-eigenschap garandeert dat er uiteindelijk iets goeds gebeurt. Met andere woorden: er bestaat een tijdstip waarop het systeem zich in de

Schermafbeelding 2023-12-08 132840.png

Schermafbeelding 2023-12-08 132959.png



Schermafbeelding 2023-12-08 133123.png


Schermafbeelding 2023-12-08 133201.png

Schermafbeelding 2023-12-08 133242.png

Schermafbeelding 2023-12-08 142543.png


Schermafbeelding 2023-12-08 142602.png

Schermafbeelding 2023-12-08 142800.png




Schermafbeelding 2023-12-08 144259.png

Schermafbeelding 2023-12-08 144315.png




Schermafbeelding 2023-12-08 144343.png


Schermafbeelding 2023-12-08 144403.png




Schermafbeelding 2023-12-08 144418.png



Schermafbeelding 2023-12-08 144433.png



Schermafbeelding 2023-12-08 144451.png




Schermafbeelding 2023-12-08 144519.png

Schermafbeelding 2023-12-08 144537.png

Schermafbeelding 2023-12-08 144614.png

Schermafbeelding 2023-12-08 144758.png



Schermafbeelding 2023-12-08 144815.png