A Decision Support System for the Maintenance of Lights of Traffic Regulation Systems

F.A. van der Duyn Schouten
A.S. Klusener
S.F.M. van Vlijmen
S.L.E. Vos de Wael

Universiteit Utrecht, Centrum voor Wiskunde en Informatica

Samenvatting

From October 1994 to June 1995 the authors executed a project, the so called 'Lampremplace' project, in cooperation with Nederland Haarlem, a Dutch constructor and supplier of traffic regulation systems. 'Lampremplace', as it is called in traffic jargon, is the maintenance of lights of traffic regulation systems. The research issue here is to find a 'good' tradeoff between the reliability of the system, and the efficiency and economy of the maintenance operation. In the project a number of maintenance strategies were developed that can be optimised and are hopefully simple in daily use. Field test of the strategies was not carried out in this project.

This is the public version of the final report of the 'Lampremplace' project. In part I of the report one finds a theoretical development that leads towards two main maintenance strategies. First, the main research questions are formulated, the assumptions under which the research was carried out are listed, and relevant statistical and computer science notions are discussed. Next, two maintenance strategies are formulated and developed. Finally, optimisation algorithms for the parameters of the strategies are formulated. A prototype that was developed in Pascal is not discussed in this version of the report.

Part II presents a formal specification, in the specification language PSF (Process Specification Formalism [MV93]), of a decision support system for the maintenance of lights of traffic regulation systems. Whereas the first part is focused on strategies and optimisation, the second part is focused on incorporating the strategies and optimisation algorithms in a complete system that can be used by parties that are responsible for the maintenance of road infrastructures.

Dit is de eindrapportage van het Lampremplaceproject dat in de periode october 1994 — juni 1995 is uitgevoerd door de auteurs voor Nederland Haarlem. In dit stuk komen aan de orde: de probleemstelling, de centrale besliskunde- en informaticanoties, de gevolgde oplossingsmethoden, en een formele specificatie van een beslissingsondersteunend systeem.

1 Inleiding

Sinds de zestiger jaren is in de wetenschappelijke literatuur vrij veel aandacht besteed aan de optimalisatie van onderhoud en vervanging van technische systemen. De toepassing van deze resultaten is tot voor kort beperkt gebleven tot zeer onderhoudsintensieve bedrijfstakken zoals de olieindustrie en defensie. Hiervoor zijn een aantal oorzaken aan te geven.

In de eerste plaats werd tot voor kort in vele bedrijfstakken het onderhoud beschouwd als een noodzakelijk kwaad. Onderhoudsbudgetten werden eenvoudigweg jaarlijks met enkele procenten verhoogd zonder dat bekeken werd of de omvang van deze budgetten in een goede verhouding stond tot de onderhoudsvraag. Dit werd in belangrijke mate versterkt door de manier waarop onderhoud veelal was georganiseerd, namelijk in een aparte onafhankelijke 'unit' die niet op 'output' werd afgerekend. Voor de onderhoudsmanager was het zaak een zo groot mogelijk budget te verwerven op grond van betrekkelijke vage noties die te maken hadden met een gevoel van veiligheid en 'werken volgens de normen'. In de olieindustrie en in het defensieapparaat is deze lijn het eerst doorbroken, hetgeen geen

verbazing mag wekken als wordt bedacht dat in een bedrijf als Shell de kosten van onderhoud die van productie inmiddels overschrijden. Daar waar in productieafdelingen steeds scherper de nadruk op efficiëntie in de productiemethoden werd gelegd, kon het niet uitblijven dat ook de onderhoudsafdelingen aan efficiëntiecriteria werden onderworpen. Dientengevolge zien we over de hele linie van het bedrijfsleven een sterk toenemende belangstelling voor onderhoudsoptimalisatie.

Een tweede oorzaak voor de slechte doorwerking van wetenschappelijke resultaten in de praktijk is gelegen in het feit dat het wetenschappelijk onderzoek tot voor kort vrijwel exclusief was gericht op zeer eenvoudige modellen die weinig aan de oplossing van praktische onderhoudsproblemen konden bijdragen. Zo is slechts de laatste vijf jaar sprake van serieuze aandacht in het wetenschappelijk onderzoek van de onderhoudsoptimalisatie voor complexe systemen bestaande uit vele componenten die elk hun eigen faalgedrag vertonen. Dit wil overigens niet zeggen dat de resultaten die verkregen zijn voor enkelvoudige systemen niet bruikbaar zouden zijn in de meer complexe praktijk. Integendeel, deze resultaten vormen veelal de elementaire bouwstenen waarmee beslissingsondersteunende systemen voor onderhoudsoptimalisatie van realistische systemen kunnen worden gebouwd, zoals in het project Lampremplace ook is gebleken.

In de context van dit project verstaan we onder 'lampremplace' het vervangen van lampen in de lichtpunten van verkeerslichtinstallaties (VLI's).

Definitie 1.1. Een VLI is het geheel aan lichtinstallaties dat het verkeer op een kruispunt helpt regelen. Een kruispunt moet hierin ruim worden opgevat; lampen die geografisch dicht genoeg bij elkaar liggen, dat wil zeggen, het overbruggen van de afstanden tussen de installaties vormt geen significante kostenpost, kunnen tot één VLI gerekend worden, terwijl zij niet door dezelfde verkeers*regel*installatie (VRI) worden aangestuurd.

In één VLI kunnen derhalve een groot aantal lampen voorkomen. De problematiek die bij lampremplace speelt is algemeen van aard, men denke bijvoorbeeld aan de lampen van de straatverlichting of de lampen in de seinen van spoorwegbeveiligingen, of nog algemener, aan een verzameling 'gelijkwaardige' componenten die regelmatig onderhoud behoeven.

Bij een onderhoudsactiviteit als lampremplace zijn twee grootheden met name van belang: de kosten van de onderhoudsactiviteit en de betrouwbaarheid van het systeem. Om economische redenen is het streven het onderhoud op een zodanige wijze uit te voeren dat tegen 'lage' kosten een 'hoge' betrouwbaarheid wordt gerealiseerd. Zo ook bij lampremplace.

Volgens Nederland Haarlem is de wijze waarop lampremplace heden ten dage plaatsvindt onvoldoende gestructureerd en begrepen. De motivatie voor dit project is gelegen in de wens van Nederland Haarlem om als aanbieder te kunnen voorzien in beslissingsondersteunende systemen voor een gestructureerde en goed onderbouwde aanpak van lampremplace [Kro94]. De doelstelling van het project was dan ook:

> Kennis opbouwen over het genereren van lampremplaceschema's die naar betrouwbaarheid en exploitatiekosten geoptimaliseerd zijn. En als het kennisniveau dit toestaat, het geven van formele specificaties voor een beslissingsondersteunend systeem en een statistisch registratiesysteem.

In het project is het volgende bereikt.

- Om structuur aan te brengen in de lampremplaceactiviteit is een vijftal strategieën geformuleerd: twee strategieën worden zeer uitgebreid behandeld en drie anderen zeer kort. Onder een strategie wordt in dit verband verstaan een vervangingsvoorschrift van de lampen in een VLI. Dit voorschrift wordt in het algemeen gekarakteriseerd door een aantal kentallen die elk via een optimalisatie numeriek moeten worden bepaald.

- De twee praktisch meest relevante van de vijf strategieën, de Indirecte Hoofdgroepering Strategie (IHG) en de 'Opportunity Based Replacement' Strategie (OBR) zijn wiskundig beschreven. Dat

wil zeggen dat een model ontwikkeld is dat uitdrukkingen geeft voor de exploitatiekosten en de betrouwbaarheid als functie van de kentallen van elk van de strategieën.

- Voor IHG is een heuristiek[1] ontwikkeld om de naar betrouwbaarheid en exploitatiekosten geoptimaliseerde kentallen voor een VLI binnen de klasse IHG te vinden. Deze heuristiek is geïmplementeerd in Pascal.
- Voor OBR is een algoritme ontwikkeld om de naar betrouwbaarheid en exploitatiekosten geoptimaliseerde kentallen voor een VLI binnen de klasse OBR te vinden. Dit algoritme is geïmplementeerd in Pascal.
- De problematiek van het beheren van meer dan één VLI wordt besproken. Er wordt een route geschetst om dit probleem aan te pakken.
- Er is een formele specificatie gegeven in PSF [MV93] die een primitief beslissingsondersteunend systeem en statistisch registratiesysteem beschrijft.
- Op basis van de specificatie en de Pascal programmatuur voor IHG en OBR is een eenvoudig prototype gebouwd.

De code van het prototype is niet opgenomen in deze versie van de rapportage. De specificatie is opgenomen in Deel II.

De drie niet uitgewerkte strategieën dienen om het probleemgebied in kaart te brengen en dienen als ijkpunt om betere strategieën tegen af te zetten. Ze worden kort behandeld bij de andere twee strategieën, IHG en OBR. Twee van deze niet uitgewerkte strategieën zou men kunnen typeren als primitieve strategieën, deze zijn theoretisch van belang geweest tijdens het onderzoek, maar hebben in onze ogen praktisch weinig relevantie; IHG en OBR zijn economisch gesproken significant beter en, naar ons idee, toch eenvoudig in het gebruik. Dat neemt niet weg dat volgens Thomassen [Haa93], deze primitieve strategieën wel degelijk gebruikt worden in de praktijk (zie de Paragrafen 4.4 en 6.5). De derde niet uitgewerkte strategie is in zekere zin een combinatie van IHG en OBR (zie Paragraaf 6.4).

Gezien de beperkte looptijd van het project, hebben wij de voorkeur gegeven aan het vinden van goed gedefinieerde en implementeerbare optimalisatiemethoden boven het uitwerken van enkele theoretische vraagstellingen die tijdens het project naar voren kwamen en die tot geavanceerdere strategieën zouden kunnen leiden. Zo wordt in Paragraaf 4.1 voor IHG een alternatieve aanpak beschreven die vooralsnog te ingewikkeld is.

Het voordeel van de gevolgde werkwijze is dat er nu een concreet fundament ligt van enkele eenvoudige en goed begrepen strategieën. Hierop kan Nederland Haarlem prototypen van systemen bouwen en ervaring opdoen in de praktijk. Parallel daaraan zou gewerkt kunnen worden aan meer geavanceerde strategieën.

Dankwoord

Wij spreken onze dank uit voor de hulp die de derde auteur ontving van Willem van Wilsem voor het uitvoeren van typ-werkzaamheden ten behoeve van Deel I en het prototype (dat niet in deze versie van de rapportage is opgenomen), en voor het maken van wijzigingen en toevoegingen aan de specificatie in Deel II.

[1] Een heuristiek is een systematische oplossingsmethode voor een wiskundig model, waarin echter — vanwege de hoge complexiteit van het wiskundig model — op sommige punten simplificaties worden aangebracht. Hierdoor is de optimaliteit van de door de heuristiek gegenereerde oplossing niet langer gegarandeerd. De prestatie van de heuristiek moet derhalve altijd worden getoetst, bijvoorbeeld door middel van statistische simulatie, danwel via vergelijking met de prestatie van primitieve strategieën.

Inhoudsopgave

1 Inleiding **1**

I Definities en Methoden **6**

2 Lampremplace **6**
2.1 Modelaannamen 7
2.2 De kostenstructuur 7

3 Inleiding besliskundige noties **8**

4 Indirecte Hoofdgroepering Strategie **13**
4.1 De algemene kostenfunctie 13
4.1.1 Vereenvoudiging van de algemene kostenfunctie 15
4.2 Eigenschappen van de kostenfunctie 16
4.2.1 Analytische eigenschappen 17
4.3 Een uitdrukking voor de betrouwbaarheid 18
4.4 Een eenvoudige instantie van IHG 18

5 Heuristiek voor IHG **19**
5.1 De heuristiek 19

6 'Opportunity Based Replacement' Strategie **20**
6.1 De algemene kostenfunctie 21
6.1.1 Vereenvoudiging van de algemene kostenfunctie 22
6.2 Eigenschappen van de kostenfunctie 23
6.3 De betrouwbaarheid van OBR 24
6.4 Een uitbreiding van OBR 24
6.5 Een eenvoudige variant op OBR 25

7 Algoritme voor OBR **25**
7.1 Het algoritme 27

8 Werken met IHG en OBR **28**
8.1 De belangrijkste gegevens 28
8.1.1 Correctie van de levensduurverdeling met een brandkental 29
8.2 De Groepsindeling 29
8.3 Sturen met de 'penalty' 30
8.4 Enkele voorbeelden 31
8.4.1 Geval 1 31
8.4.2 Geval 2 36
8.4.3 Geval 3 38
8.5 Enkele uitzonderingsgevallen 45
8.5.1 Lampen 45
8.5.2 Afwijkingen van het gegenereerde schema 48
8.5.3 Wijzigingen aan de configuratie van de VLI 48
8.5.4 Oscillatie van OBR 48

9 Aggregatiestrategieën, een aanzet **49**

10 Conclusies, aanbevelingen en vervolgstudies **49**

II Specificatie **53**

11 De specificatie in grote lijnen **53**
11.1 Executeren van de specificatie 54

12 Tijd in de specificatie **54**
12.1 Tijd 54
12.2 De soorten `Datum`, `Duur0` en `Duur1` 56
12.3 Het uitrekenen van preventieve opdrachten 56
12.4 Het bepalen van te ondernemen actie ingeval van een defect 56
12.5 De functie `onder-schema` 56
12.6 `RemplacePlanning` 57
12.7 `SchemaBeheer` 57

13 Invarianten **57**

14 Tekortkomingen van de specificatie **58**

15 Specificatie **59**
15.1 Elementaire datatypen 59
15.2 Datastructuren van `Database` 73
15.3 LRdata 79
15.4 Modulen voor simulatie en procesdefinitie 84
15.5 Processen 89

Deel I
Definities en Methoden

2 Lampremplace

In deze paragraaf wordt gedefinieerd wat de besliskundige vraagstelling is die opgelost moet worden in het geval van één enkelvoudige verkeerslichtinstallatie. Daarna worden de aannamen onder welke de strategieën zijn ontwikkeld gegeven en als laatste wordt de kostenstructuur besproken. Het lampremplaceprobleem voor een VLI is het volgende.

Probleemstelling 2.1. *Gegeven een VLI met een gegeven aantal lampen. Bedenk een strategie die gevolgd kan worden bij het preventief en correctief vervangen van deze lampen, en bedenk een handelingsvoorschrift om de strategie naar exploitatiekosten en betrouwbaarheid te optimaliseren. De optimalisatie dient uitvoerbaar te zijn op een computer en de resulterende adviezen dienen werkbaar te zijn, d.w.z., het uitvoeren van het advies moet eenvoudig en begrijpelijk zijn, zodat het uitvoeren vrijwel foutloos kan geschieden.*

In de strategieën die in dit verslag worden besproken wordt niet op lampniveau geadviseerd maar op een hoger aggregatieniveau, nl. op groepen lampen. Een strategie die op lampniveau zou adviseren is naar verwachting te gedetailleerd en daarom niet werkbaar. Een typische groepsindeling is in drieën op kleur. Dus een groep met alle 'rode' lampen[2], een groep met alle 'gele' lampen, en een groep met alle 'groene' lampen. Maar er zijn natuurlijk ook fijnere indelingen denkbaar, bijvoorbeeld, 'de groep rode lampen die alleen via een hoogwerker te bereiken is', of 'de groep groene lampen van de hoofdrichting'. In het uiterste geval kan elke lamp weer als groep worden opgevat. Bij de te ontwikkelen strategieën wordt er vanuit gegaan dat een indeling in groepen op voorhand heeft plaatsgevonden. Er is dus geen strategie beschreven die gevolgd kan worden en die tot een optimale groepsindeling én een optimaal schema leidt. Het idee is: gegeven een indeling wordt een optimaal schema berekend. Er zijn echter wel enkele richtlijnen die gevolgd kunnen worden bij het kiezen van een groepsindeling. Tevens, is het heel wel mogelijk om met een zgn. 'what-if' analyse een aantal indelingen te vergelijken. Uitputtend groepsindelingen afzoeken is daarentegen vrijwel onmogelijk, omdat het aantal mogelijke groepsindelingen van een verzameling lampen zeer sterk stijgt met de grootte van de verzameling. In Paragraaf 8 wordt dieper ingegaan op de wijze waarop een 'goede' groepsindeling kan worden gekozen.

Notatie 2.2. In het volgende wordt steeds uitgegaan van een VLI met n lampen ($n \geq 1$), genummerd $1 \ldots n$. Deze lampen zijn ingedeeld in m groepen, genummerd $1 \ldots m$. Er geldt dus $m \leq n$. Verder is het zo dat de variabele i over het algemeen slaat op de lampen (en dus loopt over $1 \ldots n$) en de variabele j slaat op de groepen (en dus loopt over $1 \ldots m$).

De verzameling lampen wordt aangeduid met S en de geïndiceerde verzameling van groepen wordt aangeduid met $(S_j)_{j=1}^m$. Het aantal lampen in groep j wordt aangeduid met n_j, dit is een afkorting voor $|S_j|$, de standaard notatie voor het aantal elementen van een verzameling. Er geldt dus dat $|S| = \sum_{j=1}^m |S_j| = \sum_{j=1}^m n_j = n$.

Voorbeeld 2.3. Als $S = \{L_1, L_2, L_3, L_4, L_5, L_6\}$ een verzameling lampen is, dan is $n = 6$. Als L_1 en L_5 in één groep zitten, zeg groep 1, en de rest in een andere groep, zeg groep 2, dan is $m = 2$, $S_1 = \{L_1, L_5\}$ en $S_2 = \{L_3, L_4, L_2, L_6\}$. Verder geldt dat $n_1 = |S_1| = 2$ en $n_2 = |S_2| = 4$. □

Betrouwbaarheid kan in het geval van lampremplace gezien worden als het aantal falende lampen per tijdseenheid. Omdat een defect direct aanleiding is tot een vernieuwing, kan de betrouwbaarheid ook geformuleerd worden als het aantal vernieuwingen per tijdseenheid.

[2] De kleur is het gevolg van de kleur van de lens, de lampen zijn identiek.

2.1 Modelaannamen

In de loop van het project is meermalen contact geweest tussen Nederland Haarlem en de uitvoerders over allerlei aspecten die samenhangen met lampremplace. Dit heeft geleid tot een aantal *model-aannamen* die enerzijds tot doel hebben het model analyseerbaar te houden. Anderzijds dienen deze modelaannamen uiteraard te stoelen op praktische overwegingen en dienen zij de werkelijkheid niet al te zeer geweld aan te doen. Uitgangspunt voor het onderzoek vormen VLI's waarvan voor élke lamp een betrouwbare schatting gegeven kan worden van de fractie per tijdseenheid dat de lamp werkelijk brandt en waarvan voor élke lamp de levensduurverdeling bij continu branden bekend is. Hieronder volgt een opsomming van de belangrijkste aannamen en vereenvoudigingen die in het model aangebracht zijn.

- Optimalisatie gebeurt per VLI. In Paragraaf 9 wordt een aanzet gegeven tot coördinatie van onderhoud van verschillende VLI's.
- Het is belangrijk om te constateren dat men een onderscheid kan maken tussen, zeg, *statische* VRI's, dit zijn VRI's die een vast patroon van lichtbeelden cyclisch afwerken, en *dynamische* VRI's, dit zijn VRI's die aan de hand van het verkeersaanbod lichtbeelden samenstellen. Het is volgens Nederland Haarlem mogelijk om voor dynamische VRI's een redelijk betrouwbare schatting te geven van de fractie per tijdseenheid dat de lamp werkelijk brandt. Deze gegevens zou men eventueel kunnen verkrijgen door meting op locatie of simulatie. Op die wijze zijn dynamische VRI's op te vatten als statische VRI's. Deze aanname draagt direct over op de notie van VLI.
- Alleen lampremplace wordt bestudeerd als te optimaliseren activiteit. Andere onderhoudsactiviteiten, zoals het reinigen van de lenzen e.d., worden niet in de beschouwing betrokken.
- Het schakelgedrag en externe factoren, zoals weer en wind, trillingen van verkeer etc., hebben vooralsnog geen invloed op de levensduur van lampen.
- Het faalgedrag van de lampen is onderling onafhankelijk. Dat wil zeggen, het faalgedrag van de ene lamp heeft geen invloed op het faalgedrag van een andere lamp.
- De levensduurverdeling van lampen volgt een Weibull- of een Erlang-verdeling. Deze aanname is mede gebaseerd op informatie van een lampenproducent. Zie voor verdere specificatie van deze aanname Paragraaf 3.
- Er wordt alleen rekening gehouden met het faalgedrag van lampen; andere onderdelen van een VLI die de werking van lampen kunnen beïnvloeden, zoals fittingen, voedingen en bedradingen, worden buiten beschouwing gelaten.
- Er wordt geen rekening gehouden met eventuele voorbereidingstijden die benodigd kunnen zijn voor het plannen en realiseren van wegafzettingen.
- Elke dag is in beginsel geschikt om een remplace-actie op uit te voeren.
- Er is altijd voldoende capaciteit beschikbaar bij de wegbeheerder voor het uitvoeren van een remplace. Dat wil zeggen, er treden geen 'wachttijden' op bij het uitvoeren van preventief of correctief onderhoud.

2.2 De kostenstructuur

De *kosten* die gemaakt worden zijn directe kosten als uurloon, huur van apparatuur en kosten van lampen. Daarnaast blijkt het nuttig te zijn ook indirecte kosten te introduceren. Deze kosten refereren

aan het optreden van ongewenste situaties, zoals een veelvuldig falen van lampen waardoor de verkeersveiligheid in gevaar komt, dit zou kunnen leiden tot aansprakelijkheidsstelling van de wegbeheerder, waardoor het verband met 'kosten' duidelijk wordt. Wel dient te worden opgemerkt dat deze indirecte kosten minder 'hard' zijn en derhalve niet als vaste invoergegevens kunnen worden beschouwd. De volgende indeling van de kosten wordt gehanteerd.

Definitie 2.4.

- Vaste kosten per remplace-actie, deze worden aangeduid met A. Dit is een basistarief waarin bijvoorbeeld het huren en reinigen van een actiewagen is verwerkt, het uurloon om heen en weer te rijden naar de VLI. Deze kosten worden dus betaald voor elke actie, ongeacht of dit een preventieve actie betreft waarin, bijv., 10 groepen vervangen worden of een correctieve waarin, bijv., twee lampen worden vervangen.
- a_j, de kosten voor het preventief vervangen van groep j. Dit zijn kosten die specifiek gemoeid zijn met het vervangen van groep j. Denk hierbij aan uurloon en materiaalkosten.
- c_i, de kosten voor het correctief vervangen van lamp i. Denk aan zaken als uurloon, materiaalkosten.
- p_j, de strafkosten (de '$p$enalties') die betaald worden als een lamp uit groep j faalt.

De strafkosten zijn de indirecte kosten zoals boven besproken, met deze kosten kan de mate van onwenselijkheid van correctief onderhoud uitgedrukt worden. Dit zijn dus geen echte kosten, zoals A, maar kunstmatige kosten. Het zal blijken dat deze kosten gebruikt kunnen worden om het het optimalisatieproces te sturen, zie Paragraaf 8.

Het feit dat er kosten A bestaan, kosten die men één keer maakt per remplace-actie ongeacht de omvang, maakt het interessant om vervangingen te combineren. Het is dus *niet* zo dat de kosten uit A getransporteerd kunnen worden naar de respectievelijke a_j's en c_i's waarbij A vervolgens op 0 wordt gesteld. Dan betaalt men zaken dubbel, althans in het model, als vervangingen gecombineerd worden. Overigens is het ook niet zo dat dit soort spanningen tussen model en werkelijkheid vermeden kan worden. Bijvoorbeeld, als VLI X alleen maar groepen heeft die met een hoogwerker te bereiken zijn, dan zitten de kosten van een hoogwerker evident in A. Als VLI Y groepen heeft die hoog én laag zitten, dan kan men overwegen de kosten van de hoogwerker voor de 'hoge' groepen te verwerken in de respectievelijke a_j's en c_i's. Combinaties van vervangingen van deze groepen geven in het model hogere kosten dan werkelijk worden uitgegeven. Dat zou er voor pleiten de kosten voor de hoogwerker uit te delen over alle groepen, of nog beter het delen van kosten veel preciezer te modelleren (zie de opmerkingen in Paragraaf 10). De theorie behoeft hier nadere uitwerking, derhalve zal men op grond van ervaring en het gezonde verstand bepaalde kosten moeten vaststellen.

3 Inleiding besliskundige noties

In deze paragraaf worden een aantal noties uit de besliskunde geïntroduceerd en geïllustreerd die in de volgende paragrafen intensief worden gebruikt. Voor een uitgebreide en diepgaande beschrijving van de noties wordt verwezen naar de literatuur [Ger89, Tij95].

Zoals in de inleiding is opgemerkt spelen de resultaten die in de wetenschappelijke literatuur zijn verkregen rond optimale planning van onderhoud en vervanging van eenvoudige systemen een belangrijke rol bij onderhoudsoptimalisatie van meer complexe systemen. In deze paragraaf introduceren we dan ook een aantal basisbegrippen en resultaten die in het vervolg intensief zullen worden gebruikt. Beschouw een technisch systeem dat continu operationeel dient te zijn en faalgedrag vertoont dat zich niet met zekerheid laat voorspellen. Wel is bekend dat het faalgedrag verband houdt met de leeftijd

van het apparaat in die zin dat de faalkans van het apparaat toeneemt naarmate het langer operationeel is geweest. Een goede beschrijving van de afhankelijkheid tussen leeftijd en faalkans wordt gegeven door een zogenaamde kansverdelingsfunctie. Een kansverdelingsfunctie F voor een gegeven systeem voegt aan ieder getal t een getal tussen 0 en 1 toe, waarbij $F(t)$ wordt geïnterpreteerd als de kans dat het systeem faalt binnen het tijdsinterval van 0 tot t, waarbij wordt aangenomen dat het systeem op tijdstip 0 nieuw was (leeftijd nul had). Duidelijk zal zijn dat een kansverdelingsfunctie dus een monotoon stijgende functie zal zijn. Immers de kans om binnen het interval $[0, t_1]$ stuk te gaan is kleiner dan de kans om binnen het interval $[0, t_2]$ stuk te gaan als $t_2 > t_1$. Een andere meer praktische interpretatie van $F(t)$ voor vaste waarde van t is de volgende. Stel dat op tijdstip 0 een groot aantal identieke systemen wordt gestart, dan zal hiervan op tijdstip t een fractie $F(t)$ kapot zijn en een fractie $(1 - F(t))$ op tijdstip t nog functioneren, aannemende dat kapotte systemen niet worden gerepareerd. Nu zijn er talloze klassen van kansverdelingsfuncties bekend en onderzocht. Echter in de context van levensduurverdelingen spelen twee klassen van kansverdelingen een belangrijke rol, namelijk de Weibull-verdelingen en de Erlang-verdelingen. Dit is niet alleen een empirisch gegeven verkregen op basis van intensieve metingen, maar hiervoor zijn ook theoretische argumenten aan te dragen. De klasse van Weibull-verdelingen wordt gekarakteriseerd door de volgende formule van de kansverdelingsfunctie:

$$F(t) = 1 - e^{-(\lambda t)^{\alpha}} \quad , t \geq 0,$$

waarbij $\alpha > 0$ en $\lambda > 0$ nader te kiezen parameters zijn. De parameter α heet de *vormparameter* en λ heet de *schaalparameter*. Nota bene, voor iedere andere keuze van α en λ heeft men te maken met een andere Weibull-verdeling. Als men derhalve weet dat de levensduur van een systeem beschreven kan worden door een Weibull-verdeling moeten vervolgens de waarden van de parameters α en λ nog worden vastgesteld; dan pas ligt de kansverdelingsfunctie vast. De klasse van Erlang-verdelingen laat zich als volgt beschrijven:

$$F(t) = \int_0^{\lambda t} \frac{s^{n-1} e^{-s}}{(n-1)!} ds,$$

waarbij ook hier twee vrije parameters voorkomen, te weten de vormparameter n, een geheel en positief getal, en $\lambda > 0$, de schaalparameter.

Bovenstaande is het algemene geval, echter voor de beschrijving van de levensduur van lampen in VLI's wordt in dit stuk uitgegaan van Weibull-verdelingen met vormparameter $\alpha > 1$ en van Erlang-verdelingen met $n \geq 2$.

Belangrijke kentallen van een kansverdelingsfunctie vormen het gemiddelde μ en de variantie σ^2, die formeel gedefinieerd worden door

$$\mu := \int_0^{\infty} (1 - F(t)) dt$$

en

$$\sigma^2 := 2 \int_0^{\infty} t(1 - F(t)) dt - \mu^2.$$

Het gemiddelde μ is een zogenaamde *locatieparameter* van de kansverdeling, als op tijdstip 0 een groot aantal identieke systemen wordt ingezet net zolang totdat alle defect zijn dan zal de gemiddelde levensduur, gemeten over alle systemen, ongeveer gelijk zijn aan μ. De variantie σ^2 is een *spreidingsparameter* van de kansverdeling, σ^2 is groter naarmate er meer variabiliteit in de waargenomen levensduren optreedt; als alle systemen precies dezelfde levensduur zouden realiseren is $\sigma^2 = 0$.

Zoals gezegd wordt zowel de klasse van Weibull-verdelingen als de klasse van Erlang-verdelingen gekenmerkt door twee parameters (α en λ voor de Weibull-verdeling en n en λ voor de Erlang-verdeling). In de praktijk wordt echter vaak een verdelingsfunctie uit een van beide klassen gekarakteriseerd door het gemiddelde en de variantie op te geven. Dit leidt tot uitspraken van de vorm: "De levensduur van dit soort gloeilampen volgt een Weibull-verdeling met een gemiddelde van 10 *uren* en een variantie

van 100 *uren*^{2}". De vraag rijst dan uiteraard met welke Weibull-verdeling we hier te maken hebben, met andere woorden welke waarden van λ en α behoren bij de gegeven $\mu_W = 10$ en $\sigma_W^2 = 100$. Het antwoord hierop wordt gegeven door de volgende relaties:

$$\mu_W = \frac{1}{\lambda}\Gamma(1 + \frac{1}{\alpha}) \tag{1}$$

en

$$\sigma_W^2 = \frac{\Gamma(1 + \frac{2}{\alpha})}{\lambda^2} - \frac{\Gamma^2(1 + \frac{1}{\alpha})}{\lambda^2}, \tag{2}$$

waarbij $\Gamma(x)$ de gammafunctie voorstelt die wordt gedefinieerd door

$$\Gamma(x) = \int_0^\infty e^{-t}t^{x-1}dt \quad , x > 0.$$

Uit de relaties (1) en (2) zijn de waarden van λ en α te bepalen als de waarden van μ_W en σ_W gegeven zijn.

Voor de Erlang-verdeling worden de verbanden tussen gemiddelde (μ_E) en variantie (σ_E^2) enerzijds en de parameters (λ en n) anderzijds gegeven door:

$$\mu_E = \frac{n}{\lambda}$$

en

$$\sigma_E^2 = \frac{n}{\lambda^2} \cdot$$

Aangezien σ^2, de maat voor de spreiding van de kansverdeling, niet dimensieloos is, wordt veelal de voorkeur gegeven aan de variatiecoëfficiënt c als maat voor de spreiding van een kansverdeling. De variatiecoëfficiënt wordt gedefinieerd door

$$c^2 = \frac{\sigma^2}{\mu^2} \cdot$$

Voor de klasse van Weibull-verdelingen en Erlang-verdelingen gelden de volgende relaties:

$$c_W^2 = \frac{\Gamma(1 + \frac{2}{\alpha})}{\Gamma^2(1 + \frac{1}{\alpha})} - 1$$

en

$$c_E^2 = \frac{1}{n} \cdot$$

Nota bene, beide zijn onafhankelijk van schaalparameter λ. In de karakterisering van levensduurverdelingen speelt verder de faalgraad (Eng. failure rate) een belangrijke rol. De faalgraad $r(t)$ representeert de intensiteit waarmee een systeem dreigt stuk te gaan als functie van zijn leeftijd en wordt gedefinieerd door

$$r(t) = \frac{F'(t)}{1 - F(t)} \cdot \tag{3}$$

In deze uitdrukking is $F'(t)$ de afgeleide van de kansverdelingsfunctie en deze wordt de kansdichtheidsfunctie genoemd en ook vaak genoteerd met $f(t)$.

De relatie tussen (3) en het begrip faalgraad wordt gelegd door de volgende observatie:

$$r(t)\Delta t \approx \frac{F(t + \Delta t) - F(t)}{1 - F(t)} \tag{4}$$

waarbij de rechterzijde van (4) kan worden geïnterpreteerd als de kans dat het systeem stuk gaat in het interval $(t, t+\Delta t]$ gegeven dat het systeem het interval $[0, t]$ heeft overleefd. Voor de Weibull-verdeling is eenvoudig na te gaan dat de faalgraad wordt gegeven door

$$r_W(t) = \alpha\lambda(\lambda t)^{\alpha-1} \quad , t \geq 0.$$

Als derhalve $\alpha > 1$, of te wel $c_W^2 < 1$, dan is de faalgraad een functie die stijgt van 0 (in $t = 0$) naar ∞ (als $t \to \infty$). We spreken in dit geval van een kansverdeling met stijgende faalgraad (Eng. increasing failure rate, d.i. IFR). Als daarentegen $\alpha \leq 1$, of te wel $c_W^2 \geq 1$, dan daalt de faalgraad van ∞ naar 0 (Eng. decreasing failure rate, DFR).

Voor de Erlang-verdeling is geen mooie uitdrukking voor de faalgraad $r_E(t)$ te geven. Wel is eenvoudig aan te tonen dat voor $n = 1$, of te wel $c_E^2 = 1$, de faalgraad een constante ($= \lambda$) is, terwijl voor $n > 1$, of te wel $c_E^2 < 1$, de faalgraad $r_E(t)$ stijgt van 0 (in $t = 0$) naar λ (als $t \to \infty$).

De meest geëigende waarde voor de parameters λ en α zullen moeten worden bepaald door schattingen met behulp van experimenten. Hierbij is het uiteraard belangrijk deze experimenten uit te voeren onder omstandigheden die vergelijkbaar zijn met de omstandigheden waaronder de lampen in de praktijk functioneren. Op grond van raadpleging van deskundigen zullen we in de context van lampremplace ervan uitgaan dat het niet de verstreken levensduur van een lamp is die bepalend is voor het faalgedrag maar de verstreken brandduur. Dit maakt voor de theoretische inbedding geen enkel verschil. Wel moet er uiteraard op worden gelet dat bij het specificeren van de relevante kansverdelingen hiermee rekening wordt gehouden. Bovendien zal een eenheid van tijd moeten worden gekozen. Het ligt voor de hand om hier het *uur* of de *dag* voor te nemen. Bijvoorbeeld omdat gegevens van lampenfabrikanten vaak in uren gespecificeerd zijn. Voor het theoretisch betoog is dat niet echt van belang mits de eenheid strikt wordt doorgevoerd.

Een en ander houdt dus in dat in de context van lampremplace $F(800) = 0.58$ betekent dat voor de lampsoort die wordt beschouwd de kans dat een exemplaar van deze lampsoort binnen 800 tijdseenheden brandduur stuk gaat 58% bedraagt.

Beschouw nu het volgende sterk gesimplificeerde probleem. Een technisch systeem heeft een levensduurverdeling $F(t)$, kansdichtheidsfunctie $f(t)$ en faalgraad $r(t)$. Als het systeem faalt wordt het onmiddellijk vervangen door een nieuw identiek systeem. Dit brengt kosten (vervangingskosten, productieverlies, aansprakelijkheidskosten etc..) met zich mee ter grootte van c' per vervanging. Een alternatief is echter het systeem preventief te vervangen (dat wil zeggen, vervanging terwijl het nog functioneert). Uiteraard brengt dit ook kosten met zich mee, zeg c, die echter lager zijn dan de correctieve vervanging, dus $c < c'$.

Een mogelijke strategie zou nu kunnen zijn: vervang het systeem bij het stukgaan of wanneer het de leeftijd T bereikt, afhankelijk van wat zich het eerste voordoet. Deze strategie wordt aangeduid als de 'age-replacement' strategie, immers het vervangingsmechanisme is volledig gebaseerd op de leeftijd van het systeem dat operationeel is. Eigenlijk is de age-replacement strategie een klasse van strategieën, omdat immers de waarde van T nog kan worden gekozen. Het is echter mogelijk binnen de klasse van age-replacement strategieën via kostenminimalisatie die waarde van T te bepalen die de laagste gemiddelde kosten per tijdseenheid oplevert. Hiertoe stellen we vast dat de gemiddelde kosten per tijdseenheid als functie van de vervangingsleeftijd T kunnen worden gerepresenteerd door:

$$C_A(T) := \frac{c(1 - F(T)) + c'F(T)}{\int_0^T (1 - F(t))dt}$$

Door nu $C_A(T)$ te minimaliseren naar T vinden we de 'economisch optimale preventieve vervangingsleeftijd' T^*. Als de faalgraad $r(t)$ een stijgende functie is kan eenvoudig worden aangetoond dat T^* gevonden kan worden als de unieke oplossing van de vergelijking

$$r(T^*) \int_0^{T^*} (1 - F(t))dt - F(T^*) = \frac{c}{c' - c} \cdot \tag{5}$$

In het algemeen zijn numerieke procedures nodig om (5) op te lossen. Merk op dat voor toepassing van de age-replacement strategie in ieder geval nodig is om de actuele leeftijd van het apparaat dat operationeel is bij te houden. Als de age-replacement strategie in de context van lampremplace op iedere individuele lamp afzonderlijk zou worden toegepast (hetgeen overigens niet verstandig lijkt omdat dan op geen enkele manier coördinatie plaatsvindt) dan zou van iedere lamp zijn verstreken brandduur moeten worden bijgehouden. Hierbij moet worden bedacht dat door tussentijdse correctieve vervangingen de brandduren binnen een overigens homogene groep toch uit elkaar zullen gaan lopen. Dit zou dus een gedetailleerd registratiesysteem vereisen.

Vanwege de bovengenoemde bezwaren zijn andere strategieën in de literatuur voorgesteld. De bekendste hiervan is de zogenaamde 'block-replacement' strategie, die ook wordt gekarakteriseerd door één nog te kiezen vrije parameter T. De block-replacement strategie schrijft voor een systeem altijd om de T tijdseenheden preventief te vervangen en daarnaast correctieve vervangingen uit te voeren wanneer een defect optreedt. Let wel, T slaat hier dus op de kalendertijd en niet langer, zoals bij age-replacement, op de verstreken brandduur van het systeem dat operationeel is. Als we ook hier kosten c en c' associëren aan een preventieve resp. correctieve vervanging dan vinden we voor de gemiddelde kosten per tijdseenheid als functie van het vervangingsinterval T de volgende uitdrukking:

$$C_B(T) = \frac{c + c'M(T)}{T} \,. \tag{6}$$

Hierbij stelt $M(t)$ voor het verwachte aantal correctieve vervangingen op het interval $[0, t]$. De functie $M(t)$, die volledig wordt bepaald door de kansverdelingsfunctie van de levensduur $F(t)$, wordt wel de vernieuwingsfunctie genoemd. Een probleem hierbij is dat voor de meeste kansverdelingsfuncties er geen eenvoudige uitdrukking voor $M(t)$ bestaat. Een uitzondering hierop is de exponentiële verdeling (Erlang-verdeling met $n = 1$). Hiervoor geldt dat de bijbehorende vernieuwingsfunctie gegeven wordt door $M(t) = \lambda t$. Maar omdat de faalgraad van deze verdelingsfunctie constant is (er treedt dus geen veroudering op) volgt onmiddellijk dat er geen optimaal vervangingsinterval bestaat, dat wil zeggen, $T^* = \infty$, hetgeen dan ook uit (6) volgt als we daarin λT substitueren voor $M(T)$ en vervolgens $C_B(T)$ naar T minimaliseren.

Het voordeel van block-replacement boven age-replacement is dat bij block-replacement geen verstreken brandduren behoeven te worden bijgehouden. Er zijn ook enkele nadelen, waarvan de complexiteit van de functie $M(t)$ overigens niet de grootste is. In de literatuur zijn inmiddels verschillende goede methoden aangedragen om $M(t)$ numeriek te bepalen voor een grote klasse van mogelijke verdelingsfuncties, zie [DS90, Tij95]. Een groter nadeel van block-replacement is echter dat het onder toepassing van deze strategie kan voorkomen dat kort voor een gepland vervangingstijdstip (op een veelvoud van T) een systeem stuk gaat en dus correctief wordt vervangen, hetgeen impliceert dat op het geplande preventieve vervangingstijdstip een nagenoeg nieuw systeem wordt vervangen. Om aan dit nadeel tegemoet te komen zijn in de literatuur verschillende varianten op block-replacement voorgesteld, ondermeer de variant die in het geval van een defect binnen een gespecificeerde afstand van een gepland vervangingstijdstip een noodmaatregel treft, die kan variëren van een noodreparatie tot het laten voortduren van het defect tot aan de eerstvolgende geplande vervanging (en de daarmee verbonden kosten van productieverlies dus te accepteren). De variant om in het geval van een defect kort voor een geplande vervanging de geplande vervanging naar voren te schuiven is, voorzover de auteurs weten, nog niet systematisch onderzocht. In het vervolg van dit verslag wordt aan deze variant enige aandacht besteed en daaruit blijkt dat, hoe voor de hand liggend deze variant ook moge zijn, de theoretische analyse ervan gecompliceerd is. Overigens merken wij hier direct maar vast op dat aan deze variant ook praktische nadelen kleven. Een belangrijk aspect van een geplande vervanging is namelijk dat deze ruim voor uitvoering bekend is en dus maatregelen voor de uitvoering ervan van tevoren kunnen worden genomen, te denken valt aan inplanning van werk voor uitvoerders, afspraken met Rijkswaterstaat aangaande wegafzetting etc. Door nu de optie in te bouwen, namelijk dat een geplande vervanging kan worden vervroegd ten gevolge van een onverwachte correctieve vervanging,

gaan bovengenoemde planningsvoordelen weer verloren. Op dit moment is nog onvoldoende duidelijk in hoeverre dit voor de lampremplace als een aanvaardbare optie moet worden beschouwd. We komen hierop terug in Paragraaf 6.4.

4 Indirecte Hoofdgroepering Strategie

In deze paragraaf wordt de Indirecte Hoofdgroepering Strategie (IHG) geformuleerd en worden uitdrukkingen gegeven voor de kostenfunctie en voor de betrouwbaarheid. Deze twee noties worden eerst zo algemeen mogelijk geformuleerd, vervolgens worden twee vereenvoudigingen aangebracht, waarvan één vooralsnog noodzakelijk is om te komen tot een hanteerbare oplossingsmethode. De IHG kan als volgt worden omschreven.

> Bij het defect raken van een lamp wordt deze direct correctief vervangen. Correctief onderhoud geeft geen aanleiding tot preventief onderhoud (in tegenstelling tot OBR, zie Paragraaf 6). Daarnaast wordt elke groep met vaste tussenpozen preventief vervangen. De tijd, dat is de kalendertijd, tussen een tweetal groepsvervangingen mag van groep tot groep verschillen maar is wel een geheel aantal malen een vast tijdsinterval genaamd de basiscyclus.

Notatie 4.1. De basiscyclus wordt aangeduid met T. Voor elke groep j is k_j, een geheel getal groter dan 0 ($k_j \in \mathbb{N} \setminus \{0\}$) dat aangeeft hoeveel basiscycli liggen tussen twee preventieve vervangingen. De naar oplopende index geordende rij van alle k_j's wordt aangeduid met de vectornotatie $\underline{k}$. Soms wordt echter geschreven $k_1, \ldots, k_m$ als dat duidelijker is, of omdat dit een makkelijke manier is om een deel van de vector aan te duiden, zoals bijvoorbeeld in $k_3, \ldots, k_6$.

We zullen nu wiskundige uitdrukkingen afleiden voor de gemiddelde kosten per tijdseenheid alsmede voor de betrouwbaarheid (faalfrequentie van de verzameling lampen in een VLI) als functie van T en $\underline{k}$. Met behulp van de wiskundige uitdrukkingen kunnen we dan een rekenschema ontwerpen waarmee de optimale waarde van T en $\underline{k}$ binnen de klasse van IHG kan worden bepaald. De kostenstructuur die gehanteerd wordt is die uit Paragraaf 2.2; een voorbeeld van het gebruik en de interpretatie van de kosten volgt hieronder.

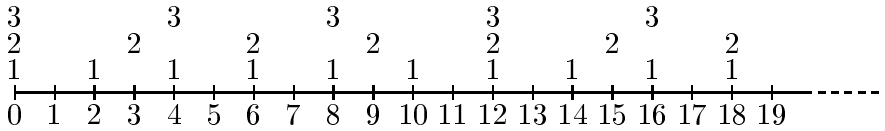
Voorbeeld 4.2. Als op een gegeven moment de groepen 1 en 5 gelijktijdig preventief worden vervangen bedragen de kosten $A + a_1 + a_5$. Als op een gegeven moment lamp nr. 75 uit groep 4 correctief moet worden vervangen bedragen de daarmee samenhangende kosten $A + p_4 + c_{75}$. □

4.1 De algemene kostenfunctie

De kostenfunctie, die een uitdrukking geeft voor de gemiddelde kosten per tijdseenheid, vormt de kern van de besliskundige vraagstelling. Deze functie drukt uit wat de gemiddelde kosten per tijdseenheid zullen zijn. Optimalisatie van de kostenfunctie, komt neer op het zoeken van die waarden van T en $\underline{k}$ waarin de functie een absoluut minimum heeft.

De kostenfunctie wordt eerst in haar meest algemene vorm gepresenteerd. Daarna worden twee vereenvoudigingen aangebracht die leiden tot een formulering die gebruikt wordt in de heuristiek in Paragraaf 5. Beginnen met een algemene formulering maakt het betoog langer dan misschien strikt noodzakelijk is. Wij denken echter dat het nuttig is voor de lezer om precies te weten wat de theoretische uitgangspunten zijn en welke vereenvoudigingen gemaakt worden. Op deze wijze kan ook worden aangegeven waar theoretische knelpunten zitten.

De kosten ('Costs') als functie van de beslissingsvariabelen T en $\underline{k}$ heeft voor IHG de volgende vorm.



Figuur 1: De groepen 1, 2 en 3 in de tijd.

$$C(T,\underline{k}) = \frac{1}{T}A\Delta(\underline{k}) + \sum_{j=1}^{m}\frac{a_j}{k_jT} + \sum_{j=1}^{m}\sum_{i\in S_j}\frac{M_i(k_jT)(A+c_i+p_j)}{k_jT}$$

waarbij (7)

$$\Delta(\underline{k}) = \sum_{\ell=1}^{m}(-1)^{\ell+1}\sum_{\{\alpha\mid\alpha\subset\{1,\ldots,m\},|\alpha|=\ell\}} kgv(k_{\alpha_1},\ldots,k_{\alpha_\ell})^{-1}$$

Deze formule is ontleend aan [Dag82], waar deze formule wordt afgeleid voor een voorraadprobleem met meerdere artikelen. Het gaat daarbij om het bepalen van zodanige bestelmomenten voor de onderscheiden artikelen dat de som van de bestelkosten en voorraadkosten zo laag mogelijk is. Aangezien ook hier de coördinatie tussen de verschillende artikelen een belangrijke rol speelt is de overeenkomst met lampremplace dermate sterk dat van dezelfde formules gebruik kan worden gemaakt. In het volgende worden eerst alle symbolen uit deze formule toegelicht en de interpretatie van de formule gegeven. Daarna wordt een voorbeeld gegeven.

Zoals eerder aangegeven, stelt T de basiscyclus voor; $\underline{k}$ is de vector gehele getallen $k_1 \ldots k_m$; voor $1 \leq j \leq m$, waarbij k_j aangeeft het aantal basiscycli tussen preventieve vervangingen van groep j; A, a_j, c_i en p_j zijn kosten zoals gedefinieerd in Paragraaf 2.2.

M_i is de vernieuwingsfunctie behorende bij de levensduurverdeling van lamp i, zie Paragraaf 3.

De uitdrukking $\{\alpha \mid \alpha \subset \{1,\ldots,m\}, |\alpha| = \ell\}$ geeft de collectie aan van alle mogelijke keuzen van ℓ getallen uit de verzameling $\{1,\ldots,m\}$. Als we de volgorde van de elementen in een verzameling α fixeren, dan kunnen we met α_k, $1 \leq k \leq |\alpha|$, de elementen van α adresseren, dit wordt gedaan in $kgv(k_{\alpha_1},\ldots,k_{\alpha_\ell})$.

De functie $C(T,\underline{k})$ bepaalt de kosten per tijdseenheid door voor iedere groep j te bepalen hoeveel kosten worden gemaakt gedurende één cyclus en dit getal te delen door de lengte van de cyclus. De eerste summand is $\frac{1}{T}A\Delta(\underline{k})$, deze berekent de basistariefkosten voor preventieve vervangingen per tijdseenheid. Hierin bepaalt $\Delta(\underline{k})$ in welk deel van de tijdstippen die een veelvoud van T zijn preventief onderhoud werkelijk plaatsvindt.

De tweede summand is $\sum_{j=1}^{m}\frac{a_j}{k_jT}$, deze representeert de som van alle extra preventieve vervangingskosten per tijdseenheid. Immers groep j wordt op het interval van de lengte k_jT precies één keer preventief vervangen tegen extra kosten a_j.

De derde summand, $\sum_{j=1}^{m}\sum_{i\in S_j}\frac{M_i(k_jT)(A+c_i+p_j)}{k_jT}$, geeft de totale correctieve vervangingskosten per tijdseenheid. Hierin is $M_i(k_jT)$ het gemiddelde aantal correctieve vervangingen van lamp i op het interval $[0,k_jT]$, startend met een nieuwe lamp i op tijdstip 0. Een vernieuwing van i kost $A+c_i+p_j$. De kosten worden vervolgens gedeeld door k_jT om de correctieve kosten per tijdseenheid te weten te komen. Voor alle lampen in groep j zijn de kosten dan $\sum_{i\in S_j}\frac{M_i(k_jT)(A+c_i+p_j)}{k_jT}$. Dit bedrag moet voor alle groepen bepaald worden en vervolgens worden de resultaten hiervan gesommeerd.
Het volgende voorbeeld dient ter illustratie van de functies in (7).

Voorbeeld 4.3. Stel er zijn drie groepen 1, 2 en 3 met $k_1 = 2$, $k_2 = 3$ en $k_3 = 4$. Het aantal groepen m is dus 3. Op tijdstip 0 zijn al deze groepen preventief vervangen, de basiscyclus heeft lengte 1. Figuur 1 beeldt uit wanneer 1, 2 en 3 preventief vervangen worden. Stel $A = 12$, $a_1 = 5$ en $a_2 = a_3 = 6$.

Laat verder elke groep twee lampen bevatten, $S_1 = \{1,2\}$, zeg $S_2 = \{3,4\}$ en $S_3 = \{5,6\}$. Het aantal lampen n is dus 6. De correctieve kosten zijn $A + c_1 + p_1 = A + c_2 + p_1 = 100$, $A + c_3 + p_2 = A + c_4 + p_2 = 50$ en $A + c_5 + p_3 = A + c_6 + p_3 = 25$. Als laatste, stel dat $M_1(k_1T) = M_2(k_1T) = 0.01$, $M_3(k_2T) = M_4(k_2T) = 0.03$, en $M_5(k_3T) = M_6(k_3T) = 0.08$.

We berekenen $C(t, \langle k_1, k_2, k_3\rangle)$ en beginnen met het bepalen van $\Delta(\langle k_1, k_3, k_3\rangle)$. Zoals te zien is in Figuur 1, moet dit $\frac{8}{12} = \frac{2}{3}$ opleveren, immers 12 kan gezien worden als de cyclustijd van het systeem en van 12 vervangmomenten worden er maar 8 echt gebruikt. Ter illustratie wordt $\Delta(\langle k_1, k_3, k_3\rangle)$ uitgeschreven.

$$\Delta(\langle k_1, k_2, k_3\rangle) =$$
$$\sum_{\ell=1}^{3}(-1)^{\ell+1} \sum_{\{\alpha \mid \alpha\subset\{1,2,3\},|\alpha|=\ell\}} kgv(k_{\alpha_1}, \ldots, k_{\alpha_\ell})^{-1} = .. =$$
$$\sum_{\{\alpha \mid \alpha\subset\{1,2,3\},|\alpha|=1\}} kgv(k_{\alpha_1})^{-1} - \sum_{\{\alpha \mid \alpha\subset\{1,2,3\},|\alpha|=2\}} kgv(k_{\alpha_1}, k_{\alpha_2})^{-1} + \sum_{\{\alpha \mid \alpha\subset\{1,2,3\},|\alpha|=3\}} kgv(k_{\alpha_1}, k_{\alpha_2}, k_{\alpha_3})^{-1}$$

De uitdrukking $\{\alpha \mid \alpha \subset \{1,2,3\}, |\alpha| = 1\}$ staat voor de verzameling van alle verzamelingen van 1 element uit $\{1,2,3\}$, dit is $\{\{1\},\{2\},\{3\}\}$. Zo staat $\{\alpha \mid \alpha \subset \{1,\ldots,m\}, |\alpha| = 2\}$ voor de verzameling van verzamelingen van 2 elementen uit $\{1,2,3\}$, dit is $\{\{1,2\},\{1,3\},\{2,3\}\}$. De laatste, $\{\alpha \mid \alpha \subset \{1,\ldots,m\}, |\alpha| = 3\}$ is dan $\{\{1,2,3\}\}$. Het geheel wordt dan

$$\sum_{\{\{1\},\{2\},\{3\}\}} kgv(k_{\alpha_1})^{-1} - \sum_{\{\{1,2\},\{1,3\},\{2,3\}\}} kgv(k_{\alpha_1}, k_{\alpha_2})^{-1} + \sum_{\{\{1,2,3\}\}} kgv(k_{\alpha_1}, k_{\alpha_2}, k_{\alpha_3})^{-1}.$$

De functie *kgv* berekent het kleinste gemene veelvoud van z'n argumenten. Bijvoorbeeld, $kgv(2) = 2$ en $kgv(2,3) = 6$. Dit resulteert in

$$\frac{1}{kgv(2)} + \frac{1}{kgv(3)} + \frac{1}{kgv(4)} - \frac{1}{kgv(2,3)} - \frac{1}{kgv(2,4)} - \frac{1}{kgv(3,4)} + \frac{1}{kgv(2,3,4)} = .. = \tfrac{2}{3}.$$

Dit is precies wat we hadden afgelezen van Figuur 1. De eerste summand wordt dus $A\Delta(\langle k_1, k_2, k_3\rangle) = 8$.
De tweede summand wordt $\sum_{j=1}^{3} \frac{a_j}{k_j} = \frac{a_1}{2} + \frac{a_2}{3} + \frac{a_3}{4} = 6$.
De derde summand, tenslotte, levert weer wat meer werk:

$$\sum_{j=1}^{3}\sum_{i\in S_j} \frac{M_i(k_j)(A+c_i+p_j)}{k_j} =$$
$$\frac{\sum_{i\in\{1,2\}} M_i(2)(A+c_i+p_1)}{2} + \frac{\sum_{i\in\{3,4\}} M_i(3)(A+c_i+p_2)}{3} + \frac{\sum_{i\in\{5,6\}} M_i(4)(A+c_i+p_3)}{4} =$$
$$\frac{M_1(2)(A+c_1+p_1)+M_2(2)(A+c_2+p_1)}{2} +$$
$$\frac{M_3(3)(A+c_3+p_2)+M_4(3)(A+c_4+p_2)}{3} +$$
$$\frac{M_5(4)(A+c_5+p_3)+M_6(4)(A+c_6+p_3)}{4} = 3.$$

In totaal geeft dit $8 + 6 + 3 = 17$. □

4.1.1 Vereenvoudiging van de algemene kostenfunctie

Er zijn twee problemen met de uitdrukking in (7).

1. Het bepalen van $\Delta(\underline{k})$ kan zeer tijdrovend zijn: bij m groepen moet $2^m - 1$ keer het kleinste gemene veelvoud van een rijtje getallen worden bepaald.

2. Het is nog niet duidelijk hoe optimalisatie van $C(T, \underline{k})$ uitgevoerd zou moeten worden. Zie ook Paragraaf 10.

Deze twee punten zullen nadere aandacht behoeven. Daarom is vooralsnog vereenvoudiging van (7) noodzakelijk. Zoals gezegd in de inleiding is, gegeven de beperkte middelen in dit project, de voorkeur gegeven aan het vinden van simpele en implementeerbare strategieën die de basis kunnen vormen voor verdere studie en evaluatiewerk in de praktijk. Optimalisatie van $C(T, \underline{k})$ zou onderwerp voor nadere studie kunnen zijn. De kostenfunctie wordt vereenvoudigd op de volgende wijze.

- De oplossingsruimte wordt beperkt door alleen naar die $\underline{k}$ te kijken waarvoor er tenminste één j is met $k_j = 1$. Voor dergelijke oplossingen is $\Delta(\underline{k}) = 1$, dit is eenvoudig aan te tonen.

Er bestaat een vermoeden dat onder vrij ruime aannamen op de kostenparameters de functie $C_{IHG}(T, \underline{k})$ zijn minimale waarde bereikt voor een paar $\langle T^*, \underline{k}^* \rangle$ met de eigenschap dat er een k_j is met de waarde 1 ($min_{j=1}^{m}(k_j^*) = 1$), zodat hier in feite niet van een beperking sprake is (let wel T^* en $\underline{k}^*$ duiden expliciete waarden aan, zijn een oplossing, uit de waarden verzameling van resp. T en $\underline{k}$). Deze bewering is echter, voorzover de auteurs weten, nooit bewezen en is dan ook niet eenvoudig te verifiëren. Merk op dat in het voorbeeld (4.3), met $\langle T = 1; k_1 = 2; k_2 = 3; k_3 = 4 \rangle$ geen alternatief $\langle T^*; k_1^*; k_2^*; k_3^* \rangle$ is te vinden waarvoor $min_{j=1}^{m}(k_j^*) = 1$ en zo dat door beide oplossingen hetzelfde vervangingsschema wordt gerealiseerd, maar dat $\langle T = 1; k_1 = 2; k_2 = 4; k_3 = 8 \rangle$ en $\langle T = 2; k_1 = 1; k_2 = 2; k_3 = 4 \rangle$ wel in hetzelfde vervangingsschema resulteren.

Verder brengen we de volgende — voor de hand liggende — vereenvoudiging aan.

- We veronderstellen dat de samenstelling van de groepen zo is gekozen dat binnen een groep alle lampen hetzelfde faalgedrag vertonen en dus alle dezelfde vernieuwingsfunctie hebben.

Dat betekent dat we voortaan met M_j aanduiden de vernieuwingsfunctie van groep j. De bijbehorende kansverdelingsfunctie wordt aangegeven met F_j, het gemiddelde met μ_j en de variantie met σ_j. Tevens voeren we de volgende afkorting in.

Notatie 4.4. De uitdrukking d_j is een afkorting voor de expressie $\sum_{i \in S_j}(A + c_i + p_j)$. Deze som geeft de totale kosten die gemoeid zijn bij het één keer correctief vervangen van elke lamp in groep j. We kunnen d_j dan ook schrijven als $n_j(A + p_j) + \sum_{i \in S_j} c_i$. Later, in Paragraaf 8, zullen we voor de uitdrukking $\sum_{i \in S_j} c_i$ ook c_j schrijven. Het zal uit de context steeds duidelijk zijn of met, bijvoorbeeld, c_1, de kosten voor groep 1 of voor lamp 1 bedoeld zijn.

De kostenfunctie die na vereenvoudiging resulteert is de volgende.

$$C_{IHG}(T, \underline{k}) = \frac{A}{T} + \sum_{j=1}^{m} \frac{a_j}{k_j T} + \sum_{j=1}^{m} \frac{M_j(k_j T) d_j}{k_j T} \tag{8}$$

Merk op dat bij het zoeken naar het minimum van (8) nooit een oplossing $\langle T = 1; k_1 = 2; k_2 = 4; k_3 = 8 \rangle$ gevonden zal worden omdat de oplossing $\langle T = 2; k_1 = 1; k_2 = 2; k_3 = 4 \rangle$ in feite hetzelfde vervangingsschema oplevert maar lagere kosten volgens (8) oplevert.

4.2 Eigenschappen van de kostenfunctie

In deze paragraaf wordt de functie $C_{IHG}(T, \underline{k})$ nader onderzocht en worden eigenschappen van deze functie afgeleid.

4.2.1 Analytische eigenschappen

De kostenfunctie (8) is separabel in $k_1, \ldots, k_m$. Dat betekent dat deze functie kan worden geschreven als een som van functies die elk slechts van T en van één k_j afhangen. Deze eigenschap vereenvoudigt het zoekproces naar het minimum van (8).

In het onderstaande lemma worden voldoende voorwaarden geformuleerd waaronder de functie $C_{IHG}(T, \underline{k})$ bij vaste keuze van $\underline{k}$ slechts één locaal minimum heeft, terwijl dit locale minimum tevens wordt aangenomen voor een eindige waarde van T. Dit minimum is dan tevens het globale minimum. Een voorwaarde die we in dit lemma o.a. nodig hebben luidt als volgt.

Voorwaarde 4.5.

$$A + \sum_{j=1}^{m} \frac{a_j}{k_j} < \sum_{j=1}^{m} \frac{d_j(1 - \frac{\sigma_j^2}{\mu_j^2})}{2k_j}$$

In practische gevallen, zegt deze voorwaarde dat de preventieve vervangingskosten niet te hoog mogen zijn in vergelijking met de correctieve vervangingskosten. Mocht deze voorwaarde niet vervuld zijn dan bestaat de mogelijkheid dat preventieve vervanging überhaupt te duur wordt.

In de lemma's die zullen volgen spelen aannamen over de vernieuwingsfuncties M_j een belangrijke rol. Deze aannamen hoeven in de praktijk niet altijd te gelden. Als niet aan deze voorwaarden wordt voldaan, bestaat de kans dat de heuristiek een oplossing vindt die niet optimaal is.

Lemma 4.6. *Stel dat voorwaarde 4.5 geldt. Zij* $g(t) = \sum_{j=1}^{m} \sum_{i \in S_j} c_i m_j(k_j t)$, *hierin is* m_j *de afgeleide van de vernieuwingsfunctie* M_j. *Stel dat de afgeleide van* $g(t)$ *de volgende eigenschap heeft: er bestaan* $a, b \in \mathbb{R}^+$ *zodanig dat* $a < b$, $g'(t) \leq 0$ *voor* $t \in [0, a] \cup [b, \infty)$ *en* $g'(t) > 0$ *voor* $t \in (a, b)$. *Dan heeft* $C_{IHG}(T, \underline{k})$ *bij vaste* $\underline{k}$ *één locaal minimum dat voor eindige waarde van* T *wordt aangenomen.*

Voor optimalisatie naar een k_j (bij vaste T dus) is maar een deel van de kostenfunctie relevant, de functie is immers separabel, daarom wordt de functie C_{IHG} uit (8) hier iets anders geschreven, nl. zó dat de delen die relevant zijn voor de optimalisatie naar k_j geïsoleerd zijn:

$$C_{IHG}(T; k_1, \ldots, k_m) = \frac{A}{T} + \sum_{j=1}^{m} h_j(T, k_j) \tag{9}$$

waarbij

$$h_j(T, k_j) = \frac{a_j + M_j(k_j T) d_j}{k_j T}, \qquad 1 \leq j \leq m. \tag{10}$$

Beschouw nu de volgende voorwaarde.

Voorwaarde 4.7.

$$a_j < \frac{d_j(1 - \frac{\sigma_j^2}{\mu_j^2})}{2}, \qquad \textit{met } 1 \leq j \leq m.$$

Het volgende lemma kan worden bewezen.

Lemma 4.8. *Stel dat voorwaarde 4.7 geldt én voor alle groepen* j $(1 \leq j \leq m)$ *is de functie* $m_j(t)$, *dit is de afgeleide van* $M_j(t)$, *continu en strikt stijgend op het interval* $[0, \infty)$. *Definieer* $h_j(T, 0) = \infty$. *Dan bestaat er bij vaste* $T > 0$ *één eindige waarde voor* $k_j \geq 1$ *zodanig dat* $h_j(T, k_j - 1) \leq h_j(T, k_j) \leq h_j(T, k_j + 1)$.

Voor optimalisatie naar T is het plezierig als op voorhand een zo klein mogelijk interval bepaald kan worden waarbinnen de optimale waarde van T moet liggen. Er zijn twee lemma's die uitspraken doen over de grenzen van een dergelijk interval. Eerst volgen echter enkele definities.

Definitie 4.9. T_j is díe waarde voor $t \geq 0$ waarvoor $h_j(t,1)$ minimaal is. Zij $\underline{1}$ de vector ter lengte m met elk element gelijk 1. T_e is díe waarde voor $t \geq 0$ waarvoor $C_{IHG}(t,\underline{1})$ minimaal is. Fixeer een zekere $\underline{k}$. $T^*_{\underline{k}}$ is díe waarde voor $t \geq 0$ waarvoor $C_{IHG}(t,\underline{k})$ minimaal is.

De volgende lemma's zeggen dat een optimum $T^*_{\underline{k}}$ ligt tussen $min^m_{j=1}(T_j/k_j)$ en T_e.

Lemma 4.10. *Zij $\underline{k} \in (\mathbb{N} \setminus \{0\})^m$. Dan geldt: $T^*_{\underline{k}} \geq min^m_{j=1}(T_j/k_j)$.*

Lemma 4.11. *Zij $\underline{k} \in (\mathbb{N} \setminus \{0\})^m$ en neem aan dat de afgeleiden van de vernieuwingsfuncties M_j allen stijgend zijn, dat er tenminste één is die strikt stijgend is, en dat $C_{IHG}(T^*_{\underline{k}},\underline{k}) \leq C_{IHG}(T_e,\underline{1})$. Dan geldt: $T^*_{\underline{k}} \leq T_e$.*

4.3 Een uitdrukking voor de betrouwbaarheid

In deze paragraaf wordt een functie gedefinieerd die een maat geeft voor de betrouwbaarheid, ofwel '*R*eliability'.

$$R_{IHG}(T,\underline{k}) = \frac{\sum_{j=1}^m n_j M_j(k_j T)}{k_j T} \tag{11}$$

$M_j(k_jT)$ geeft aan het verwachte aantal correctieve vervangingen op een onderhoudsinterval van de lengte k_jT voor een lamp uit groep j. Door te vermenigvuldigen met n_j, dit is het aantal lampen in groep j, te delen door de lengte van dit interval, en te sommeren over alle groepen hebben we een uitdrukking voor het gemiddelde aantal correctieve vervangingen per tijdseenheid.

4.4 Een eenvoudige instantie van IHG

Een eenvoudige instantie van IHG is de volgende strategie.

> Het gehele systeem wordt om de T tijdseenheden in zijn geheel vervangen. Bij het defect raken van een lamp dient deze direct te worden vervangen; er wordt bij correctief vervangen geen gebruik gemaakt van de mogelijkheid andere lampen preventief te onderhouden.

Deze strategie is niets anders dan de IHG strategie waarbij op voorhand alle $k_j = 1$ worden gekozen. De kostenfunctie en betrouwbaarheidsfunctie vereenvoudigen in dit geval tot:

$$C'(T) = \frac{A + \sum_{j=1}^m a_j + \sum_{j=1}^m M_j(T) d_j}{T}$$

en

$$R'(T) = \frac{\sum_{j=1}^m n_j M_j(T)}{T}.$$

In de praktijk wordt deze strategie gebruikt [Haa93], het is derhalve nuttig haar hier te formaliseren.

5 Heuristiek voor IHG

In beginsel is het bepalen van het minimum van de functie (8) een standaard probleem uit de wiskundige optimalisatietheorie. Echter als het aantal groepen groter wordt (zeg groter dan 10) zal het vinden van het minimum zeer veel computertijd vergen. Daarom wordt hier een heuristiek voorgesteld die, weliswaar niet gegarandeerd het minimum oplevert, maar waarvan op grond van zowel theoretische argumenten alsook van uitvoerig experimenteel onderzoek kan worden gezegd dat deze tot bevredigende resultaten leidt, in die zin dat in korte tijd een oplossing wordt gegenereerd die een functiewaarde oplevert die dicht tot zeer dicht bij het gezochte minimum ligt[3]. Het probleem dat opgelost dient te worden is het volgende[4].

> Vind die $T \geq 0$ en geheeltallige k_j, $1 \leq j \leq m$, waarvoor de functie $C_{IHG}(T, k_1, \ldots, k_m)$ een minimale waarde heeft.

De kern van de heuristiek is het om en om minimaliseren van C_{IHG} naar T en naar $\underline{k}$. Het minimaliseren naar $\underline{k}$ wordt gedaan door voor iedere k_j, $1 \leq j \leq m$, een minimalisatie uit te voeren (op grond van de boven geconstateerde separabiliteit van (8)).

5.1 De heuristiek

De heuristiek wordt nu op informele wijze beschreven. In de onderstaande tekst staat tussen dubbele aanhalingstekens ".." een actie of een test informeel beschreven. De $\circ$ staat voor sequentiële compositie van acties of processen, $a \circ b$ betekent eerst a dan b. De $+$ staat voor de keuze, dus $a + b$ staat voor uitvoeren van a óf b. De $\tau \longmapsto a$ staat voor een door een test τ afgeschermde actie of proces. Als de test τ faalt kan a niet uitgevoerd worden. Haakjes, [en], worden gebruikt voor het groeperen van acties en processen. Deze notatie stamt uit de procesalgebra [BW90]. Verder staat $\underline{1}$ voor een vector ter lengte m waarin ieder element 1 is.

Initialisatie =
"Lees kosten en statistische gegevens" $\circ$ [
"voorwaarde 4.5 voor $\underline{k} = \underline{1}$ geldt niet of voorwaarde 4.7 geldt niet" $\longmapsto$ *Faal*
+
"voorwaarde 4.5 voor $\underline{k} = \underline{1}$ geldt en voorwaarde 4.7 geldt" $\longmapsto$
"Bepaal de T_j's en T_e (zie Definitie 4.9). Zij $T = min_{j=1}^{m}(T_j)$. Bepaal voor elke groep j een k_j door $h_j(T, k_j)$ te optimaliseren naar k_j: als $h_j(T,1) \leq h_j(T,2)$, kies $k_j = 1$, anders, kies de laagste $k_j \geq 2$ die voldoet aan $h_j(T, k_j - 1) \geq h_j(T, k_j) \leq h_j(T, k_j + 1)$. Optimaliseer $C_{IHG}(T, \underline{k})$ naar T. Laat dit als optimale argumentwaarde $T^*_{\underline{k}}$ opleveren. " $\circ$ [
$C_{IHG}(T^*_{\underline{k}}, \underline{k}) \leq C_{IHG}(T_e, \underline{1}) \longmapsto$ *Optimalisatie*$_{\underline{k}}(T^*_{\underline{k}}, \underline{k})$
+
$C_{IHG}(T^*_{\underline{k}}, \underline{k}) > C_{IHG}(T_e, \underline{1}) \longmapsto$ *Optimalisatie*$_{\underline{k}}(T_e, \underline{k})$]]

Optimalisatie$_{\underline{k}}(T, \underline{q})$ =
"Bepaal voor elke groep j een r_j door $h_j(T, r_j)$ te optimaliseren naar r_j: als $h_j(T,1) \leq h_j(T,2)$, kies $r_j = 1$, anders, kies de laagste $r_j \geq 2$ die voldoet aan $h_j(T, r_j - 1) \geq h_j(T, r_j) \leq h_j(T, r_j + 1)$.
Vergelijk voor iedere groep j de kosten voor de nieuwe waarden r_j en de oude waarden q_j: als $h_j(T, r_j) \leq h_j(T, q_j)$, dan $k_j := r_j$, anders $k_j := q_j$." $\circ$ [
$\underline{q} = \underline{k} \longmapsto$ *Termineer_succesvol*$(T, \underline{k})$
+
$\underline{q} \neq \underline{k} \longmapsto$ *Optimalisatie*$_T(\underline{k})$]

[3] Voor een klein aantal groepen, zeg tot 5, is de rekentijd enkele miniuten. De rekentijd ontwikkeld zich grofweg lineair in het aantal groepen.

[4] Het probleem is gericht op het minimaliseren van de exploitatiekosten. Voldoen aan een vastgestelde betrouwbaarheid is ook van belang, echter dit is een veel eenvoudiger proces, dat beschreven wordt in Paragraaf 8. Zie ook Paragraaf 4.3.

$Optimalisatie_T(\underline{k}) =$
"voorwaarde 4.5 voor $\underline{k}$ geldt niet." $\longmapsto$ *Faal*
+
"voorwaarde 4.5 voor $\underline{k}$ geldt." $\longmapsto$
"Optimaliseer $C_{IHG}(T, \underline{k})$ naar T." $\circ$ $Optimalisatie_{\underline{k}}(T, \underline{k})$

Voor een nadere onderbouwing van de heuristiek wordt verwezen naar [Vos95]. Hier vermelden wij slechts dat het starten van de heuristiek op verschillende manieren kan. Uit numerieke experimenten is gebleken dat de aangegeven manier goed lijkt te voldoen. Praktijktesten en verder onderzoek zullen mogelijk aanleiding kunnen geven tot zwaardere voorwaarden waaraan op de invoer van de heuristiek moet voldoen.

Opmerking 5.1. Terminatie van het algoritme is niet bewezen. De volgende constateringen zouden gebruikt kunnen worden bij de bestudering hiervan. De waarde van de kostenfunctie zoals die wordt bepaald in de processen $Optimalisatie_{\underline{k}}$ en $Optimalisatie_T$ is niet-stijgend in het doorlopen aantal malen dat deze processen worden uitgevoerd. De bovengrens op de optimale oplossing wordt tijdens executie dus steeds scherper. Verder begrenst voorwaarde 4.5 feitelijk de waarden van de k_j's. Dit betekent dat het optimalisatieproces óf termineert met een oplossing óf termineert met een foutmelding omdat bijvoorbeeld niet aan voorwaarde 4.5 wordt voldaan.

6 'Opportunity Based Replacement' Strategie

De 'Opportunity Based Replacement' strategie (OBR) is een strategie waarin preventieve remplaceacties gestuurd worden door correctieve vervangingen. In OBR wordt dus van 'opportunities' gebruik gemaakt: als men toch naar het plein moet om een kapotte lamp te vervangen, dan vervangt men meteen die groepen die 'oud' zijn. Wat oud is is precies de vraag die het optimalisatiealgoritme voor OBR, zie Paragraaf 7, beantwoordt. Een opportunity is de gelegenheid voor een groep om vervangen te worden zonder daarvoor gestraft te worden. Bijvoorbeeld, in groep k faalt een lamp, er moet dus vervangen worden, een andere groep, zeg j is 'oud', dus j wordt meevervangen. Het falen van de lamp uit k is een opportunity voor j, voor groep k is het falen géén opportunity! De strategie OBR kan als volgt worden gespecificeerd.

> Bij het defect raken van een lamp wordt deze direct correctief vervangen. Correctief onderhoud kan aanleiding geven tot preventief onderhoud (in tegenstelling tot IHG, zie Paragraaf 4). Indien correctief onderhoud in het systeem wordt uitgevoerd wordt groep j preventief vervangen indien minstens t_j tijdseenheden verstreken zijn sinds groep j voor het laatst preventief vervangen werd.

Merk op dat, in tegenstelling tot IHG, er bij OBR minder sprake is van planning van preventief onderhoud. Bij IHG kan (lang) van te voren precies worden vastgesteld wanneer preventief onderhoud aan groep j moet worden uitgevoerd, nl. om de k_jT tijdseenheden. Bij OBR kan men slechts vaststellen dat binnen een tijdsbestek ter lengte t_j tijdseenheden na een preventieve vervanging geen preventief onderhoud zal worden uitgevoerd aan groep j. Als echter t_j tijdseenheden verstreken zijn dan is het wachten op de eerste lamp die binnen de VLI stuk gaat. Merk op dat bij OBR de tijd tussen twee opeenvolgende preventieve groepsvervangingen niet langer vastligt, maar mede door het 'toeval' wordt bepaald. Omdat lampen niet perfect zijn wordt er vroeg of laat altijd een defect gegenereerd. Desalniettemin zal de tijd tussen twee vervangingen een cyclus genoemd worden, zij het dat de lengte van de cyclus stochastisch bepaald is.

De kostenstructuur voor OBR is de kostenstructuur uit Paragraaf 2.2. IHG en OBR hebben derhalve dezelfde kostenstructuur. Het gebruik van de kosten in OBR is echter meer gedifferentieerd dan in IHG. In IHG was slechts sprake van correctief en preventief onderhoud. In OBR ligt de

zaak genuanceerder, correctief onderhoud kan een opportunity genereren voor preventief onderhoud. Daarbij is van belang om te bepalen in welke groep een lamp faalt, die groep betaalt namelijk de strafkosten en de basiskosten. Laat in groep j een lamp i falen op tijdstip t. Dan is van belang om te weten of t voor t_j valt of erna. In het eerste geval betaalt groep j namelijk $A + c_i + p_j$ en in het tweede geval $A + a_j + p_j$. Hieruit volgt het nut van het wachten op een opportunity. Stel $t > t_k$. Als een preventieve vervanging van groep k wordt gecombineerd met een correctieve vervanging van lamp i uit groep $j \neq k$, dan bedragen de totale kosten in de gevallen $t < t_j$ en $t \geq t_j$ respectievelijk $A + c_i + p_j + a_k$ en $A + a_j + p_j + a_k$. Als echter deze acties separaat worden uitgevoerd bedragen de kosten voor deze gevallen respectievelijk $A + c_i + p_j + A + a_k$ en $A + a_j + p_j + A + a_k$. Zie ook Paragraaf 6.4.

6.1 De algemene kostenfunctie

In deze paragraaf wordt de algemene kostenfunctie gegeven, opnieuw representerend de gemiddelde kosten per tijdseenheid. In Paragraaf 6.1.1 worden vervolgens twee aannamen gedaan die een vereenvoudiging van de kostenfunctie mogelijk maken.

Er wordt uitgegaan van een situatie waarin het systeem nieuw is op tijdstip 0. Zoals gezegd, wordt groep j preventief vervangen op het eerste moment waarop een lamp in het systeem faalt nadat er minstens t_j tijdseenheden verstreken zijn sinds de groep voor het laatst vervangen werd. Hoewel het falen van een lamp op zich geen vreugdevolle zaak is spreken we toch van een 'opportunity' voor andere groepen, omdat hierdoor zich een gelegenheid voordoet preventief onderhoud met correctief onderhoud te combineren.

Voor de analyse van een groep, zeg j, moet het ontstaan van opportunities door lampen buiten groep j gemodelleerd worden. Een manier hiervoor is een opportunity genererend proces te introduceren dat het gedrag van alle andere groepen $k \neq j$ benadert. Over dit proces wordt het volgende aangenomen.

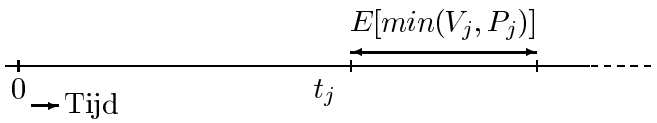
Aanname 6.1. *Als benadering voor het opportunity genererende proces voor groep j wordt een Poisson proces P_j verondersteld met intensiteit π_j.*

Deze aanname houdt in dat op een willekeurig tijdstip de tijd tot de eerste opportunity voor groep j een kansvariabele is met een exponentiële verdeling met parameter π_j. In Paragraaf 3 is de exponentiële verdeling geïntroduceerd als een bijzonder geval van de Erlang-verdeling ($n = 1$).

Het is wellicht goed op te merken dat Aanname 6.1 niet inhoudt dat we nu veronderstellen dat de levensduur van een individuele lamp een exponentiële verdeling heeft. Immers het optreden van een opportunity is een proces dat bepaald wordt door alle lampen in gezamenlijkheid, in die zin dat het optreden van een opportunity bepaald wordt door de *eerste* lamp uit het totaal van lampen in een VLI die stuk gaat. Er zijn theoretische argumenten aan te dragen die de veronderstelling dat dit tijdsinterval een exponentiële verdeling volgt aannemelijk maken ook al hebben de individuele lampen een Weibull-verdeling, Erlang-verdeling of andere verdeling.

De beslissingsvariabelen die bepaald moeten worden zijn de t_j's en de π_j's. De π_j's zullen langs analytische weg moeten worden bepaald. Deze parameters zijn overigens wel afhankelijk van de gevonden waarden van de t_j's. Een gevolg van aanname 6.1 is dat groepen afzonderlijk en met dezelfde kostenfunctie kunnen worden geanalyseerd. In het navolgende wordt derhalve de analyse beperkt tot groep j, $1 \leq j \leq m$. Uit Paragraaf 4.1.1 worden de volgende notaties overgenomen: d_j, de kosten voor het eenmalig correctief vervangen van elke lamp uit groep j, en M_j de vernieuwingsfunctie van de lampen van groep j.

De gemiddelde kosten per tijdseenheid ten gevolge van het onderhoud aan groep j worden gere-



Figuur 2: De tijdas van groep j.

presenteerd door:

$$C_j(t_j) = \frac{d_j M_j(t_j)}{t_j + E[min(V_j, P_j)]} + \frac{Pr(P_j < V_j) a_j}{t_j + E[min(V_j, P_j)]} + \frac{Pr(P_j \geq V_j)(A + a_j + p_j)}{t_j + E[min(V_j, P_j)]}. \quad (12)$$

In Formule (12) stellen P_j en V_j twee kansvariabelen voor. P_j de tijd representeert vanaf t_j tot het optreden van de eerste opportunity na t_j, terwijl V_j representeert de tijd vanaf t_j en het eerste moment waarop een lamp uit groep j uitvalt.

De ratio achter Formule (12) is de volgende. In de noemer staat weer de verwachte cycluslengte, dit is de verwachte tijd tussen twee opeenvolgende groepsvervangingen. Deze cyclustijd duurt in ieder geval t_j waarna moet worden gewacht op hetzij een defect in groep j, dit duurt nog V_j, hetzij op een defect buiten groep j, dit duurt nog P_j. Totaal neemt een cyclus naar verwachting dus $t_j + E(min(P_j, V_j))$. Aangezien zowel P_j en V_j stochastische variabelen zijn moet hier de verwachtingswaarde, de zgn. *E*xpectation worden genomen. De cyclus tijd voor groep j is in Figuur 2 uitgebeeld.

In de tellers staan weer de kosten. De eerste term representeert de verwachte kosten ten gevolge van correctieve vervangingen tot aan moment t_j. De kosten die daaraan nog moeten worden toegevoegd hangen af van wat zich het eerst voordoet: een opportunity doordat een lamp buiten groep j faalt, of een defect binnen groep j. Als een opportunity zich als eerste aandient, de kans daarop is $Pr(P_j < V_j)$, dan kan van het 'free-rider' principe gebruik worden gemaakt en bedragen de kosten van de preventieve vervanging van groep j slechts a_j, immers de vaste kosten A worden uitgespaard terwijl ook de strafkosten aan een andere groep worden toegerekend. Als daarentegen een lamp uit groep j stuk gaat voordat zich een opportunity heeft aangediend moet de volle mep aan preventieve vervangingskosten $A + a_j$, en strafkosten p_j, in rekening worden gebracht.

De algemene vorm van de kostenfunctie is nu behandeld. In de volgende paragraaf worden enkele aannames gemaakt die tot een formulering leiden van de kostenfunctie die gebruikt zal worden in het algoritme dat wordt beschreven in Paragraaf 7.

6.1.1 Vereenvoudiging van de algemene kostenfunctie

De volgende twee aannamen worden gedaan.

Aanname 6.2. *De kans dat één en dezelfde lamp uit een willekeurige groep j op het interval $[0, t_j]$ twee of meer maal defect gaat is verwaarloosbaar klein.*

Aanname 6.3. *Lampen die defect zijn gegaan in het interval $[0, t_j]$ worden op tijdstip t_j als nieuw beschouwd.*

Overigens moet via simulatie worden getoetst of Aannamen 6.1, 6.2, 6.3 in de praktijk houdbaar zijn. Hierover wordt gerapporteerd in [Vos95]. Deze vereenvoudigende aannamen maken het mogelijk de kostenfunctie in (12) explicieter weer te geven; uitwerking hiervan wordt verwezen naar [Vos95]:

$$C_j(t_j) = \frac{d_j M_j(t_j) + a_j + (A + p_j)(1 - \pi_j L_j(t_j))}{t_j + L_j(t_j)} \tag{13}$$

waarbij

$$L_j(t_j) = \int_0^\infty [\overline{F_j}(x) F_j(t_j) + \overline{F_j}(x + t_j)]^{n_j} e^{-\pi_j x} dx. \tag{14}$$

In het volgende wordt wederom de vectornotatie gebruikt voor de vector van beslissingsvariabelen: $\underline{t} := \langle t_1, \ldots, t_m \rangle$ en $\underline{\pi} := \langle \pi_1, \ldots, \pi_m \rangle$. De totale kosten per tijdseenheid, over alle groepen gesommeerd wordt dus:

$$C_{OBR}(\underline{t}) = \sum_{j=1}^{m} C_j(t_j). \tag{15}$$

In de volgende paragraaf wordt een uitspraak gedaan over het bestaan van minima van deze functie.

6.2 Eigenschappen van de kostenfunctie

Het is verleidelijk om te menen dat — net als bij IHG — de totale kostenfunctie voor OBR, zoals in (13) gerepresenteerd, opnieuw separabel is in de beslissingsvariabelen t_j. Met andere woorden, het is verleidelijk te menen dat $C_{OBR}(\underline{t})$ de som is van functies $C_j(t_j)$ die elk voor zich van slechts één t_j afhangen. Dit is echter schijn. Immers de kostenfunctie van groep j, dit is $C_j(t_j)$, hangt mede af van π_j, de intensiteit waarmee opportunities voor groep j zich aandienen. Deze intensiteit is echter op zijn beurt weer afhankelijk van alle overige beslissingsvariabelen t_k met $k \neq j$. Dit kunnen we bijvoorbeeld als volgt inzien. Als de t_k alle klein worden gekozen (dit is een conservatief preventief vervangingsbeleid) dan zullen er weinig defecten optreden en derhalve zullen ook weinig opportunities voor groep j zich aandienen, met andere woorden, π_j zal ook klein zijn. Desalniettemin zijn er over het bestaan van minima van $C_j(t_j)$ enkele nuttige uitspraken te doen. De kostenfunctie, $C_j(t_j)$ blijkt één minimum te hebben dat voor eindige waarde van t_j wordt aangenomen, mits een aantal voorwaarden gelden. Definieer $L_j(\infty)$ en $\eta_j(t_j)$ als volgt.

$$L_j(\infty) := \lim_{t_j \to \infty} L_j(t_j) = \int_0^\infty [\overline{F_j}(x)]^{n_j} e^{-\pi_j x} dx \tag{16}$$

$$\eta_j(t_j) := \frac{d_j m_j(t_j) - (A + p_j)\pi_j \ell_j(t_j)}{1 + \ell_j(t_j)} \tag{17}$$

In de laatste uitdrukking is m_j de afgeleide van M_j en ℓ_j de afgeleide van L_j uit (14).

Lemma 6.4. *De functie $C_j(t_j)$ heeft één minimum dat voor eindige waarde van t_j wordt aangenomen indien de volgende drie voorwaarden gelden:*

1. $m_j(0) = 0$;
2. $\eta_j(t)$ *is continu en stijgend op* $[0, \infty)$;
3. $a_j + A + p_j < L_j(\infty)(\frac{d_j}{\mu_j} + (A + p_j)\pi_j) + \frac{1}{2} d_j (1 - \frac{\sigma_j^2}{\mu_j^2})$.

Onder de voorwaarden van Lemma 6.4 kan tevens worden aangetoond dat de functie $\eta_j(t)$ de functie $C_j(t)$ in minima van onderen snijdt en in maxima van boven. Als $\eta_j(t)$ stijgend is op $[0,\infty)$ volgt hieruit dat $C_j(t)$ geen maxima kan hebben en dus ook niet meer dan één locaal minimum.

Aan de eerste eis is altijd voldaan ingeval van een Weibull- of Erlang-verdeling (behalve als de vormparameter n gelijk 1 is, dus ingeval van een exponentiële verdeling). Of de tweede en de derde eis gelden hangt af van de kostenparameters in samenhang met de waarden van π_j, σ_j, μ_j, en van de toestand van het optimalisatieproces; de tweede en derde eis moeten dynamisch gecontroleerd worden. De tweede eis is echter enigszins problematisch omdat dynamisch controleren moeilijk is. Daarom formuleren we een algoritme voor de bepaling van het minimum van de kostenfunctie onder de volgende

Hypothese 6.5. *Fixeer een willekeurige groep j. Op intuïtieve gronden is het aannemelijk dat een globaal minimum van C_j links ligt van de verwachte levensduur, μ_j, van lampen uit j.*

De intuïtie is gebaseerd op de gedachte dat bij realistische strafkosten p_j, het vervangen vóór de gemiddelde levensduur verstreken is profijtelijk is. Het algoritme dat in Paragraaf 7 wordt gerepresenteerd is op deze hypothese gebaseerd. Ook hier geldt dat validatie via simulatie geboden is, tevens zou nadere studie van voorwaarde 2 uit Lemma 6.4 ondernomen kunnen worden. In [Vos95] is een resultaat afgeleid voor het geval dat de resterende levensduur van de lampen exponentieel verdeeld is. In dat geval worden de eisen aan η afgezwakt.

6.3 De betrouwbaarheid van OBR

In deze paragraaf worden uitdrukkingen voor de betrouwbaarheid geformuleerd. Definieer wederom n_j als het aantal lampen in groep j. Stel dat de t_j's en de π_j's bepaald zijn. Dan wordt het aantal vernieuwingen per tijdseenheid gegeven door de volgende functie:

$$R_{OBR}(\underline{t}) = \sum_{j=1}^{m} \lambda_j(t_j) \tag{18}$$

waarbij

$$\lambda_j(t_j) = \frac{n_j M_j(t_j) + 1 - \pi_j L_j(t_j)}{t_j + L_j(t_j)}. \tag{19}$$

$\lambda_j(t_j)$ is dus een uitdrukking voor het verwachte aantal defecten per tijdseenheid in groep j. Deze uitdrukkingen worden in het algoritme in Paragraaf 7 gebruikt om de intensiteit π_j van de Poisson processen te bepalen. Men zal zien dat in het algoritme om en om naar t_j's en π_j's geoptimaliseerd wordt. In het algoritme worden nieuwe π'_j's berekend door de volgende sommen uit te rekenen:

$$\pi'_j := \sum_{k=1,k\neq j}^{m} \lambda_k(t_k). \tag{20}$$

6.4 Een uitbreiding van OBR

In deze paragraaf wordt een variant van OBR besproken. Als een preventieve vervanging van groep j wordt gecombineerd met een correctieve vervanging van lamp i die ook in groep j zit bedragen de kosten $A + a_j + p_j$, terwijl bij separate uitvoering de kosten $2A + a_j + p_j + c_i$ hadden bedragen. Hieruit kan niet zonder meer worden geconcludeerd dat wachten op het stukgaan van een lamp altijd voordeliger zou zijn. Immers door preventieve vervanging zonder dat een lamp stuk is, kunnen met name de strafkosten worden gedrukt. Deze beschouwing zou er voor kunnen pleiten de OBR strategie als volgt uit te breiden.

Bij defect raken van een lamp wordt onmiddellijk correctief vervangen. Vervang groep j preventief T_j tijdseenheden na de vorige preventieve vervanging. Als correctief onderhoud moet worden uitgevoerd dan vervangen we groep j echter preventief gelijktijdig mee als de tijd verstreken sinds de vorige preventieve vervanging tenminste t_j bedraagt. Hierbij is $t_j \leq T_j$.

In zekere zin combineert deze strategie aspecten van OBR en IHG, en intuïtief gesproken lijkt het een sterke strategie. Het is vooralsnog echter onduidelijk of deze strategie zich prettig wiskundig laat modelleren en of er snelle algoritmen voor de optimalisatie kunnen worden geformuleerd.

6.5 Een eenvoudige variant op OBR

Een vereenvoudiging van OBR is de 'laissez-faire' strategie: bij een defect wordt correctief onderhoud gepleegd, er wordt geen preventief onderhoud uitgevoerd. Alhoewel een dergelijke strategie de verkeersveiligheid niet ten goede lijkt te komen wordt ze volgens Thomassen [Haa93] wel degelijk toegepast.

Voor 'laissez-faire' kunnen we de volgende uitdrukkingen geven voor de kosten per tijdseenheid resp. de uitval per tijdseenheid.

$$C'' = \sum_{i=1}^{n} \frac{A + c_i}{\mu_i}$$

$$R'' = \sum_{i=1}^{n} \frac{1}{\mu_i}$$

Merk op dat de notie van groepen bij 'laissez-faire' niet relevant is en derhalve over lampen gesommeerd wordt. Tevens spelen strafkosten geen rol meer want er wordt alleen correctief onderhoud gepleegd en er wordt niet geoptimaliseerd.

7 Algoritme voor OBR

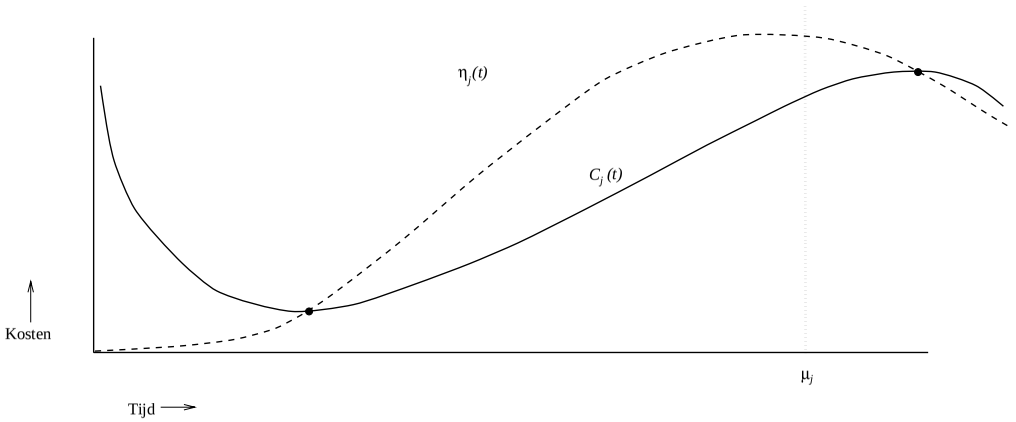
In deze paragraaf wordt een algoritme beschreven waarmee waarden voor de beslissingsvariabelen $\underline{t}$, en voor $\underline{\pi}$ kunnen worden vastgesteld. Het algoritme lijkt enigzins op IHG. Bij IHG wordt na de initialisatie geoptimaliseerd naar $\underline{k}$ en T. Bij OBR wordt na de initialisatie geoptimaliseerd naar $\underline{t}$ en $\underline{\pi}$.

In de initialisatie fase van OBR wordt zowel voor $\underline{t}$ als voor $\underline{\pi}$ een waarde vastgesteld. Zoals gezegd wordt daarna om en om naar $\underline{t}$ en $\underline{\pi}$ geoptimaliseerd. De optimalisatie stopt als $\underline{t}$ voldoende geconvergeerd is. Net als bij IHG is convergentie niet gegarandeerd. Het blijkt bijvoorbeeld dat het algoritme soms oscilleert, d.w.z. om en om met twee, volgens het convergentie criterium afwijkende oplossingen, komt. Voor een nadere bespreking van oscillatie, zie Paragraaf 8.5.4.

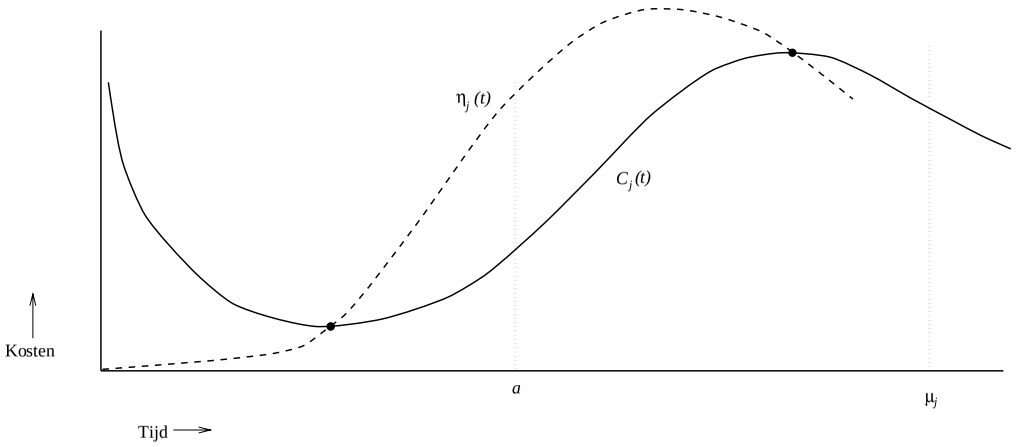
Bij de optimalisatie naar $\underline{t}$ spelen $C_j(t)$ en $\eta_j(t)$ een belangrijke rol: een optimum is een punt waar $\eta_j(t)$ de functie $C_j(t)$ van onder snijdt. Twee veelvuldig voorkomende situaties zijn geschetst in Figuur 3 en 4. Merk op dat $\eta_j(t)$ in beide afwijkt van de eis in Lemma 6.4, immers $\eta_j(t)$ is niet stijgend op $[0, \infty)$. Dit is een situatie die in de praktijk vaker zal voorkomen, ook op $[0, \mu_j]$. Het gevolg is dat er mogelijk meer lokale minima liggen op $[0, \mu_j]$ waardoor het zoekproces ingewikkelder wordt in de programmatuur dan op grond van Lemma 6.4 verwacht zou mogen worden: één bisectie op $[0, \mu_j]$ voldoet niet. In het algoritme wordt eerst gezocht naar een punt a waar geldt $\eta_j(a) > C_j(a)$, waarna bisectie op [0,a] plaats vindt; een dergelijke a is aangegeven in figuur 4.

Om oscillatie te voorkomen, voor zover mogelijk, wordt de 'ontwikkeling' van $\underline{t}$ respectievelijk $\underline{\pi}$ geremd door in een nieuwe waarde de oude waarde te verdisconteren. Dit gaat als volgt.

$$\mathit{nieuwe\_waarde} := \omega \cdot \mathit{oude\_waarde} + (1 - \omega) \cdot \mathit{berekende\_nieuwe\_waarde} \tag{21}$$



Figuur 3: Een *schets* van de grafieken van C_j en η_j. De functie C_j heeft hier één minimum en geen maximum op $[0, \mu_j]$.



Figuur 4: Een *schets* van de grafieken van C_j en η_j. De functie C_j heeft hier één minimum en één maximum op $[0, \mu_j]$.

De factor ω wordt de dempingsfactor[5] genoemd, er geldt: $\omega \in [0,1)$. Als $\omega = 0$ wordt gekozen, is er geen demping; als ω dicht bij 1 wordt gekozen is er sterke demping.

7.1 Het algoritme

Voor de betekenis van de symbolen $+$, $\circ$, $\longmapsto$, '[' en ']' wordt verwezen naar Paragraaf 5.1. Fixeer een groep j en bepaalde waarden voor de kosten A, c_i, a_j, p_j, en de parameters van de levensduurverdelingen van de groepen.

Initialisatie =
"Lees kosten en statistische gegevens." $\circ$ [
"de eerste eis van Lemma 6.4 geldt" $\longmapsto$
"Bepaal de optimale vervangings t_j voor alle groepen bij $\pi_j = \infty$, zie appendices van [Vos95]. D.w.z. vind voor elke groep j díe t_j waarvoor $\frac{d_j M_j(t_j)+a_j}{t_j}$ minimaal is." $\circ$
"Bepaal voor elke groep j de initiële π_j als volgt. $\pi_j := \sum_{k=1,k\neq j}^{m} \frac{n_k M_k(t_k)}{t_k}$" $\circ$
$Optimalisatie_{\underline{t}}(\underline{t}, \underline{\pi}, eerste\ keer)$
$+$
"de eerste eis van Lemma 6.4 geldt niet" $\longmapsto$ *Faal*]

$Optimalisatie_{\underline{t}}(\underline{t}, \underline{\pi}, keer)$ =
"de derde eis van Lemma 6.4 geldt" $\longmapsto$ $Optimalisatie'_{\underline{t}}(\underline{t}, \underline{\pi}, keer)$
$+$
"de derde eis van Lemma 6.4 geldt niet" $\longmapsto$ *Faal*

$Optimalisatie'_{\underline{t}}(\underline{t}^{oud}, \underline{\pi}, keer)$ =
$[\eta_j(\mu_j) - C_j(\mu_j) \geq 0 \longmapsto$
"Zie Figuur 3. In dit geval heeft $C_j(t_j)$ n minima en $n-1$ maxima ($n \in \{1,2,\dots\}$) op $[0,\mu_j]$. Het minimum wordt bepaald door met een bisectie-methode een nulpunt van de functie $\eta_j(t_j) - C_j(t_j)$ te bepalen op het interval $[0,\mu_j]$."
$+$
$\eta_j(\mu_j) - C_j(\mu_j) < 0 \longmapsto$
"Zie Figuur 4. In dit geval heeft $C_j(\mu_j)$ n minima en n maxima ($n \in \{0,1,2,\dots\}$) op $[0,\mu_j]$. Bepaal een punt a in $[0,\mu_j]$ waar geldt dat η_j groter is dan C_j. Om a te bepalen wordt $\eta_j(\frac{\mu_j}{2}) - C_j(\frac{M_j}{2})$ berekend; indien dit positief is, is men klaar. Anders wordt de waarde van $\eta_j - C_j$ in de middens van nu ontstane intervallen $[0, \frac{\mu_j}{2}]$ en $[\frac{\mu_j}{2}, \mu_j]$ bepaald. Is in één van deze middens $\eta_j - C_j > 0$, dan is men klaar. Anders worden deze twee intervallen weer opgedeeld etc. Het algoritme faalt als dit proces niet in een opgegeven aantal stappen eindigt." $\circ$
"Bepaal het minimum van C_j door met een bisectie-methode het nulpunt van de functie $\eta_j - C_j$ te bepalen op het interval $[0,a]$. "] $\circ$
[$keer = eerste\ keer \longmapsto Berekenen_{\underline{\pi}}(\underline{t}, \underline{\pi})$
$+$
$keer \neq eerste\ keer \longmapsto$
"voor alle $1 \leq j \leq m$, $t_j^{nieuw} := \omega_1 t_j^{oud} + (1-\omega_1)t_j$" $\circ$
["$\underline{t}^{oud}$ en $\underline{t}^{nieuw}$ voldoende geconvergeerd" $\longmapsto$
$Termineer\_succesvol(\underline{t}^{nieuw})$
$+$
"$\underline{t}^{oud}$ en $\underline{t}^{nieuw}$ onvoldoende geconvergeerd" $\longmapsto$
$Berekenen_{\underline{\pi}}(\underline{t}^{nieuw}, \underline{\pi})$
]

[5] Een andere, mogelijk meer gangbare, benaming is relaxatiefactor.

$Berekenen_{\underline{\pi}}(\underline{t}, \underline{\pi}^{oud}) =$

"Bereken een nieuwe vector Poisson parameters $\underline{\pi}^{nieuw}$:
$\pi_j^{nieuw} = \omega_2 \pi_j^{oud} + (1-\omega_2) \sum_{k=1,k\neq j}^{m} \lambda_k(t_k, \pi_k^{oud})$ " $\circ$ [
[$\underline{\pi}^{oud}$ en $\underline{\pi}^{nieuw}$ voldoende geconvergeerd $\longmapsto$
$Optimalisatie_{\underline{t}}(\underline{t}, \underline{\pi}^{nieuw}, volgende\ keer)$
+
$\underline{\pi}^{oud}$ en $\underline{\pi}^{nieuw}$ onvoldoende geconvergeerd $\longmapsto$
$Berekenen_{\underline{\pi}}(\underline{t}, \underline{\pi}^{nieuw})$
]

Convergentie van de oplossingen voor $\underline{t}$ treedt op als $\sum_{j=1}^{m} \left| \frac{t_j^{oud} - t_j^{nieuw}}{t_j^{nieuw}} \right| < c_1 \cdot m$ met $c_1 > 0$, bijvoorbeeld $c_1 = 0,1\%$. Vermenigvuldiging met m dient om 'even kritisch' te zijn bij veel groepen als bij weinig groepen. De convergentie bij π wordt op vergelijkbare wijze getest: $\sum_{j=1}^{m} \left| \frac{\pi_j^{oud} - \pi_j^{nieuw}}{\pi_j^{nieuw}} \right| < c_2 \cdot m$ met $c_2 > 0$, bijvoorbeeld $c_2 = 0,1\%$.

8 Werken met IHG en OBR

Stel men wil een remplaceschema genereren voor een zekere VLI. Om te beginnen zijn dan een aantal globale gegevens nodig en een aantal gegevens per lamp. Vervolgens kiest men een strategie en een groepsindeling. Daarna zal in een iteratief proces een schema bepaald worden dat optimaal is in termen van kosten per tijdseenheid en gegeven de gewenste betrouwbaarheid. In deze paragraaf worden een aantal zaken besproken die een belangrijke rol spelen bij het werken met de strategieën IHG en OBR. Zo komen eerst aan de orde: de gegevens die van belang zijn, richtlijnen voor het kiezen van een groepsindeling, en het sturen van het schemageneratieproces met strafkosten. Daarna worden een aantal voorbeelden besproken. De paragraaf wordt besloten met het bespreken van een aantal uitzonderingssituaties.

8.1 De belangrijkste gegevens

Als basis voor schemageneratie dienen in eerste instantie een aantal gegevens op lampniveau. Immers, de lampgegevens worden gebruikt om tot een groepsindeling te komen en zijn uiteraard mede bepalend voor het vervangingsschema. De gegevens die van elke lamp i, $1 \leq i \leq n$, bekend moeten zijn, zijn de volgende.

- De locatie van lamp i, vooralsnog wordt alleen onderscheid gemaakt tussen 'hoog' en 'laag'.
- Een kental dat aangeeft welke fractie van een tijdspanne lamp i werkelijk brand, notatie b_i. Het kental b_i wordt het brandkental van lamp i genoemd.
- Een specificatie van de levensduurverdeling van i bij continu branden.

  Weibull α_i en λ_i, resp. de vorm- en schaalparameter van de Weibull-verdeling.

  Erlang n_i en λ_i, resp. de vorm- en schaalparameter van de Erlang-verdeling.

Op basis van bovenstaande gegevens wordt een groepsindeling gekozen en worden de initiële[6] strafkosten p_j vastgesteld, preventieve groepsvervangingskosten a_j en de correctieve vervangingskosten c_j

[6] De strafkosten worden eventueel later, tijdens het generatieproces, aangepast als de verhouding tussen kosten en uitval niet strookt met de wensen van de beheerder.

(zie Notatie 4.4) berekend, dan wel vastgesteld[7]. Te zamen bestaat de invoer van de algoritmiek voor IHG en OBR dan uit de volgende componenten.

- A, het basistarief van een remplace-operatie.
- n_j, het aantal lampen in groep j.
- a_j, de kosten voor het preventief vervangen van groep j.
- c_j, de gezamelijke kosten voor het eenmalig correctief vervangen van elke lamp in groep j.
- p_j, de strafkosten van een correctieve vervanging van een lamp uit groep j.
- b_j, een kental dat aangeeft welke fractie van een tijdsspanne lampen in groep j werkelijk branden. Het kental b_j wordt het brandkental van groep j genoemd[8].
- Specificatie van de levensduurverdeling van de lampen van groep j bij continu branden[8].

  Weibull α_j en λ_j, resp. de vorm- en schaalparameter van de Weibull-verdeling.

  Erlang n_j en λ_j, resp. de vorm- en schaalparameter van de Erlang-verdeling[9].

Uit deze invoergegevens kunnen een aantal noties per groep worden afgeleid, met name, de voor b_j gecorrigeerde levensduurverdeling (zie hieronder) en de daarbij behorende vernieuwingsfunctie $M_j(t)$, de verwachting μ_j en de variantie σ_j^2. Als dit gedaan is kan met de behulp van de algoritmiek voor IHG of OBR, zoals beschreven in resp. Paragraaf 5 en 7, een remplaceschema worden gegenereerd. Een remplaceschema voor IHG bestaat uit een waarde voor T en $\underline{k}$. Een remplaceschema voor OBR bestaat uit een waarde voor $\underline{t}$ en $\underline{\pi}$. Tevens worden berekend de verwachte gemiddelde kosten per tijdseenheid (met en zonder strafkosten) en de verwachte gemiddelde uitval per tijdseenheid.

8.1.1 Correctie van de levensduurverdeling met een brandkental

Een brandkental van een lamp specificeert de fractie van een tijdsspanne dat de lamp werkelijk brandt. Het brandkental, zeg b, is derhalve een getal uit het reële interval $[0, 1]$. Als $b = 1$ dan brandt de lamp constant en als $b = 0$ dan is de lamp altijd uit. Omdat we aannemen dat het aantal schakelingen van een lamp zijn levensduur niet beïnvloedt, zie Paragraaf 2.1, is er een eenvoudige relatie tussen de levensduur verdeling bij continu branden ($b = 1$) en de levensduurverdeling bij niet continu branden ($0 \leq b < 1$). Voor een Weibull-verdeling gekarakteriseerd door α en λ wordt de voor b gecorrigeerde Weibull-verdeling gekarakteriseerd door α en $b\lambda$. Voor een Erlang-verdeling gekarakteriseerd door n en λ wordt de voor b gecorrigeerde Erlang-verdeling gekarakteriseerd door n en $b\lambda$.

8.2 De Groepsindeling

Bepalend voor de groepsindeling zijn twee karakteristieken. In de eerste plaats dienen, afgezien van individuele correctieve vervangingen, de lampen binnen één en dezelfde groep een min of meer identiek faalgedrag te vertonen. Aangezien het faalgedrag in belangrijke mate bepaald wordt door de levensduurverdeling en de brandduur, impliceert dit dat lampen met onderling afwijkende levensduurverdelingen of significante onderlinge verschillen in brandduur meestal niet in één en dezelfde groep

[7] In principe kunnen de vervangingskosten a_j en c_j berekend worden aan de hand van het aantal lampen, hun type en hun locatie; men zou dan bijvoorbeeld kunnen stellen $c_j = \sum_{i \in S_j} c_i$. De beheerder kan echter van de uitkomst van deze berekening afwijken om bijvoorbeeld extra kosten onder te brengen.

[8] Merk op dat de lampen in een groep als identiek worden beschouwd in het model.

[9] De notatie n_j is ook in gebruik als aanduiding voor het aantal lampen in groep j. Uit de context zal echter steeds duidelijk zijn welke betekenis bedoeld wordt.

zullen worden opgenomen. Als vuistregel zou men kunnen hanteren dat lampen van verschillende typen niet in één en dezelfde groep geplaatst mogen worden.

In de tweede plaats is een karakteristiek van een zinvolle groepsindeling gelegen in de kostenstructuur. Veelal is bij vervanging van alle lampen in een groep sprake van vaste kosten (kosten die zouden moeten worden gemaakt als slechts één lamp uit de groep werd vervangen) en variabele kosten (kosten die lineair toenemen met het aantal te vervangen lampen). Denk bij vaste kosten, bijvoorbeeld, aan het inzetten van een hoogwerker voor hoog hangende lampen, danwel aan de kosten van een wegafzetting. Bij variabele kosten ware te denken aan materiaal en arbeidskosten. In het licht hiervan, zal een groepsindeling mede bepaald worden door optredende vaste kosten, in die zin dat lampen die bepaalde vaste kosten gemeenschappelijk hebben eerder voor samenvoeging in dezelfde groep in aanmerking komen dan lampen die zulke vaste kosten niet gemeenschappelijk hebben.

Groepsindeling is niet triviaal, er zijn dan ook een aantal opmerkingen over te maken. Zoals boven in de eerste karakteristiek is gesteld, dienen lampen in een groep een min of meer identiek faalgedrag te vertonen. Twee lampen met hetzelfde brandkental en met dezelfde levensduurverdeling hebben, onder de aannamen van Paragraaf 2.1, hetzelfde faalgedrag en zijn volgens de eerste karakteristiek geschikte kandidaten om in één groep op te nemen. Echter het zal mogelijk voorkomen dat lampen met afwijkend brandkental in een groep geplaatst worden. Als onderdeel van een groep krijgen deze lampen geforceerd hetzelfde faalgedrag. Uit veiligheidsoverwegingen lijkt het wijs om het hoogste brandkental van de lampen in zo'n groep te nemen als brandkental voor de groep. Een dergelijke 'grovere' groepsindeling kan men verkiezen om de rekentechnische analyse wat te vereenvoudigen of om het aantal groepen te verkleinen en daardoor de uitvoering te vereenvoudigen, maar men betaalt wel een prijs; in een groep met vaak brandende en minder vaak brandende lampen worden de laatste te vaak vervangen. Desalniettemin kan dit verlies aan 'goede' lampen opwegen tegen andere kosten bij een fijnere groepsindeling. Een richtlijn valt hier vooralsnog niet te geven.

Het lijkt een mogelijkheid om in een groep de faalkarakteristieken van de individuele lampen te respecteren. In zekere zin worden op deze wijze de faalkarakteristieken uitgemiddeld. Een dergelijke aanpak heeft in elk geval als nadeel dat de rekentijd van de optimalisatie langer wordt; op zich hoeft dat niet bezwaarlijk te zijn omdat de optimalisatie slechts eenmalig wordt uitgevoerd. Voor IHG kan een dergelijke verfijning zonder technische problemen worden doorgevoerd, de tweede vereenvoudiging in Paragraaf 4.1.1 wordt simpelweg niet doorgevoerd. Voor OBR lijkt een dergelijke aanpassing grotere technische consequenties te hebben.

8.3 Sturen met de 'penalty'

Een vraag die zich aandient is hoe om te gaan in een optimalisatieproces met de functies die de gemiddelde kosten per tijdseenheid en de gemiddelde uitval per tijdseenheid beschrijven. Voor IHG zijn dit $C_{IHG}(T, \underline{k})$ en $R_{IHG}(T, \underline{k})$ en voor OBR $C_{OBR}(\underline{t})$ en $R_{OBR}(\underline{t})$. Immers, bij de optimalisatie van C_X, $X \in \{IHG, OBR\}$, wordt geen expliciete grens voor het aantal correctieve vernieuwingen in acht genomen. De werkwijze die wordt voorgesteld is de volgende. De gebruiker voert iteratief optimalisaties van C_X uit en controleert of R_X 'voldoende dicht' bij een door hem of haar gekozen waarde ligt. Zo ja, dan wordt gestopt en is het optimale preventieve vervangingsschema gevonden. Zo nee, dan wordt het fictieve deel in de kostenstructuur, nl. de strafkosten p_j, aangepast. Verhogen van de strafkosten leidt tot hogere minimale kosten maar tot een lager aantal correctieve vervangingen per tijdseenheid, en omgekeerd leidt verlagen van de strafkosten tot lagere kosten en hogere uitval per tijdseenheid.

Voor de kosten A, a_j en c_j zal het over het algemeen niet moeilijk zijn om bepaalde initiële waarden vast te stellen op grond van gegevens uit de praktijk. Voor de strafkosten ligt dat moeilijker. Er worden in voorwaarde 4.5 en 4.7, en in Lemma 6.4 relaties gegeven tussen de kosten, waarmee men een ondergrens voor de p_j's zou kunnen bepalen. Het prototype test slechts op deze voorwaarden. Een echt product zou kunnen melden wat de ondergrens is gegeven A en de a_j's en c_j's. Bij gebrek

groep	#	a	c	p	b	type	α	λ
1	10	100	100	1500	0.5	Weibull	7.61	7.44E-4

Tabel 1: De gegevens van groep 1.

T	1179 uur	1.6 maanden
k_1	1	
Kosten met straf	*FL.* 0.19/uur	*FL.* 1664/jaar
Kosten[10]	*FL.* 0.17/uur	*FL.* 1489/jaar
Uitval	0.000016 defecten/uur	0.14 defecten/jaar

Tabel 2: Optimalisatie met IHG van de gegevens in Tabel 1.

aan dergelijke intelligentie in het prototype is een vuistregel de strafkosten hoger maken dan elk van de andere kosten. Significant hogere strafkosten zullen meestal geen probleem opleveren bij de optimalisatie, er is dan zo goed als zeker aan de voorwaarden voldaan.

Opgemerkt dient te worden dat IHG en OBR niet in dezelfde mate stuurbaar zijn. Bij IHG laat de betrouwbaarheid zich vrijelijk sturen door de strafkosten, bij OBR is dat niet het geval. Dit is als volgt in te zien, bij IHG, zie (8), leidt verhogen van p_j tot het verhogen van d_j en derhalve tot het sterker stijgen van de kostenfactor $M_j(k_jT)d_j$. M_j is een continue en strikt stijgende functie door de oorsprong. Bij toename van p_j schuift het optimum k_jT naar links en omgekeerd. Bij OBR is de situatie anders, preventief onderhoud wordt slechts gepleegd als er een lamp defect is. Het defect gaan van een lamp is een stochastisch fenomeen dat slechts in beperkte mate te beïnvloeden valt door schuiven van de t_j's. Beschouw Figuur 2 (pag. 22). Voor een groep j is de verwachte lengte van een cyclus $t_j + E[min(V_j, P_j)]$. Verhogen van p_j leidt tot een lagere t_j maar ook tot een toename van $E[min(V_j, P_j)]$. De gehele cyclus wordt korter bij het stijgen van p_j, maar het effect is gering en bij elke volgende verhoging van p_j neemt het effect af.

8.4 Enkele voorbeelden

In deze paragraaf worden enkele rekenvoorbeelden gegeven op basis van gefingeerde gegevens en op basis van realistische gegevens van VLI's. In het laatste geval wordt gebruik gemaakt van gegevens die Nederland Haarlem heeft verkregen van een wegbeheerder en een lampleverancier (Philips). In alle gevallen zijn de optimalisaties uitgevoerd met het prototype. We bekijken nu eerst twee eenvoudige gefingeerde gevallen.

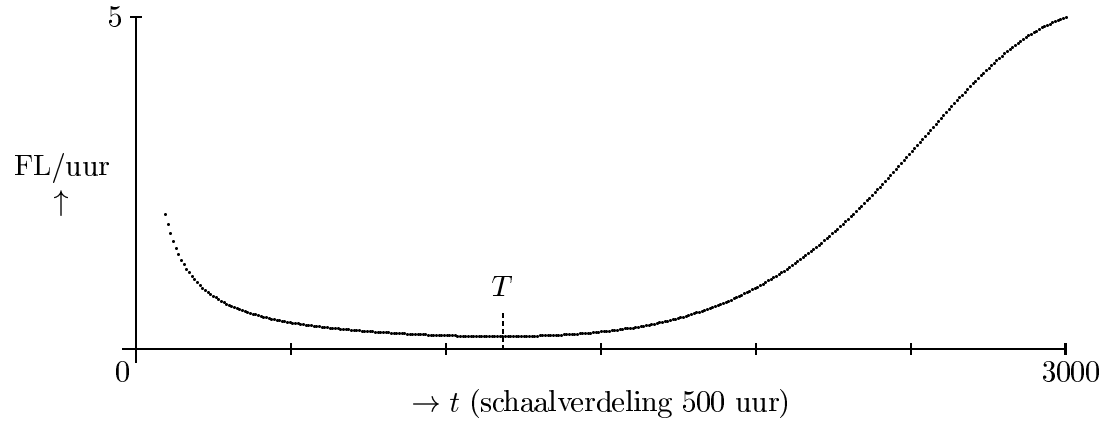
8.4.1 Geval 1

In deze sectie worden enkele eenvoudige voorbeelden gegeven waarvan op intuïtieve gronden duidelijk is wat de uitkomst van de optimalisatie grofweg zal zijn. In alle beschouwde gevallen geldt dat de basiskosten A gelijk *FL.* 100 zijn. In Tabel 1 is een groep, die we in het vervolg groep 1 zullen noemen, gekarakteriseerd.

De resultaten van optimalisatie met IHG zijn weergegeven in Tabel 2. De resultaten zijn goed te verklaren als we de levensduurverdeling van groep 1 beschouwen, zie Figuur 5. Merk op dat deze levensduurverdeling geconstrueerd is; een dergelijk sterke kromming zal in praktijk eerder uitzondering dan regel zijn. Het ligt voor de hand dat lampen van groep 1 vervangen moeten worden voor de faalkans omhoog gaat, zeg voor 1500 uur. Als we nu de kostenfunctie naar T bij vaste $\underline{k}$ beschouwen zien we dat deze zijn optimum voor deze waarde heeft liggen, zie Figuur 6.

[10] Optimalisatie levert, onder medeneming van strafkosten, waarden voor T en $\underline{k}$ respectievelijk $\underline{t}$ en $\underline{\pi}$. Gegeven deze waarden kan men de waarde van de kostenfunctie ná optimalisatie bepalen met strafkosten en zonder strafkosten (kies voor alle groepen j de p_j gelijk 0). De eerste kosten zijn hier aangeduid als 'Kosten met straf' en de tweede als 'Kosten'.

Figuur 5: De levensduurverdeling van de lampen uit groep 1 uit Tabel 1.



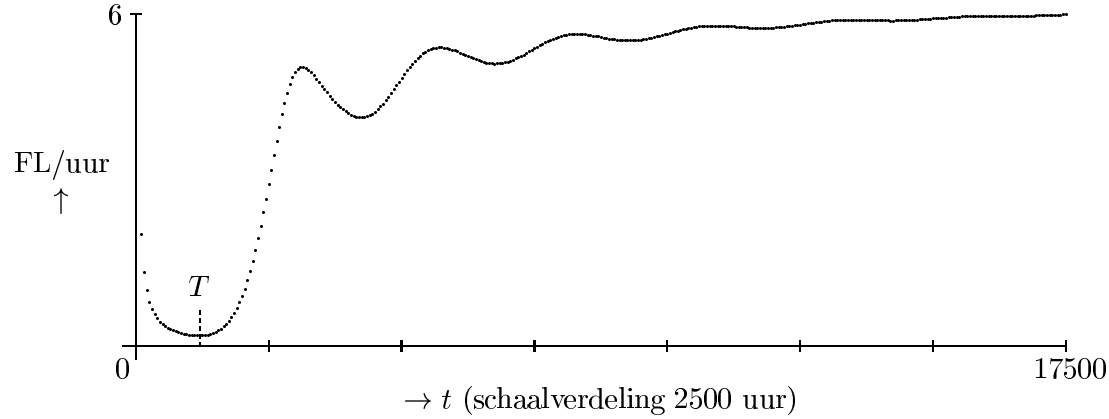
Figuur 6: De kostenfunctie naar T bij vaste $\underline{k}$ ($k_1 = 1$) van groep 1 uit Tabel 1.

Voor de volledigheid wordt in Figuur 7 de kostenfunctie voor groep 1 uit Tabel 1 voor een groot tijdsinterval afgebeeld. Merk op dat de kostenfunctie enkele locale minima heeft maar dat het globale minimum het eerste optimum is gerekend vanaf 0. Tevens is hier duidelijk te zien hoe de kostenfunctie van onderen nadert naar een asymptoot. Er kan worden aangetoond dat de kostenfuncties van IHG bij vaste $\underline{k}$ altijd dit gedrag vertonen [Vos95], mits aan Voorwaarde 4.5 is voldaan.

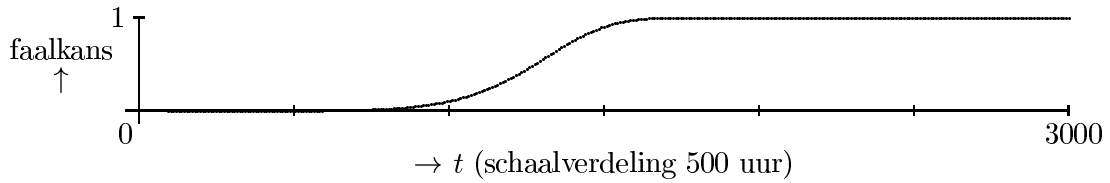
In het vervolg zullen we de effecten van wijzigingen aan de gegevens van groep 1 bestuderen; een andere groep introduceren, groep 2; en daarna de combinatie van beide groepen bekijken.

De twee wijzigingen aan de gegevens van groep 1 zijn de volgende: $b_1 = 1$ en $p_1 = 3000$. Ten eerste dus het verhogen van b_1 van 0.5 naar 1. Het moge duidelijk zijn dat de faalkans nu eerder omhoog schiet, vergelijk Figuur 5 en 8, en dat ten gevolge daarvan de optimale T ongeveer de helft zal bedragen van de oude T, zie Figuur 9 en Tabel 3.

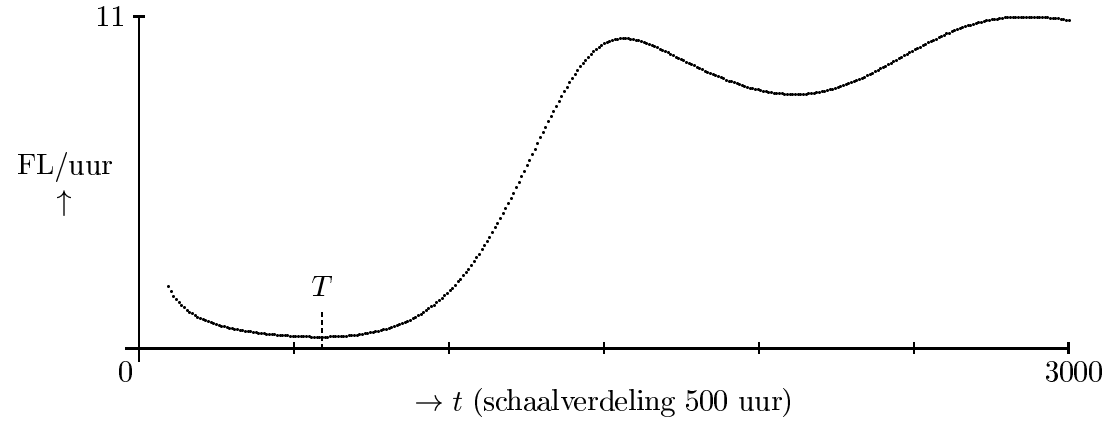
Ten tweede wordt ook p_1 verhoogd, van 1500 naar 3000. Ook dit brengt de T verder omlaag, maar minder sterk. Dit komt omdat de strafkosten een onderdeel van een som vormen terwijl het



Figuur 7: Asymptotisch gedrag van de kostenfunctie van groep 1. Merk op dat [0, 17500] het interval voor T is, dit i.t.t. de andere figuren in dit voorbeeld.



Figuur 8: De levensduurverdeling van de lampen uit groep 1 (Tabel 1) *bij* brandkental $b_1 = 1$.



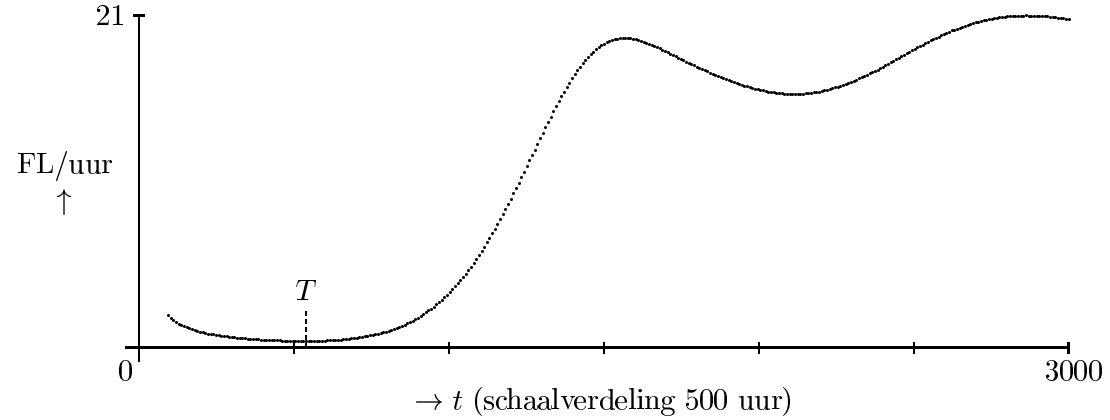
Figuur 9: De kostenfunctie naar T bij vaste $\underline{k}$ ($k_1 = 1$) van groep 1 (Tabel 1) *bij* brandkental $b_1 = 1$.

T	589 uur	3.5 weken
k_1	1	
Kosten met straf	*FL.* 0.39/uur	*FL.* 3416/jaar
Kosten	*FL.* 0.34/uur	*FL.* 2978/jaar
Uitval	0.000032 defecten/uur	0.28 defecten/jaar

Tabel 3: Optimalisatie van groep 1 met IHG bij brandkental $b_1 = 1$.

T	540 uur	3.2 weken
k_1	1	
Kosten met straf	*FL.* 0.43/uur	*FL.* 3767/jaar
Kosten	*FL.* 0.37/uur	*FL.* 3241/jaar
Uitval	0.000018 defecten/uur	0.16 defecten/jaar

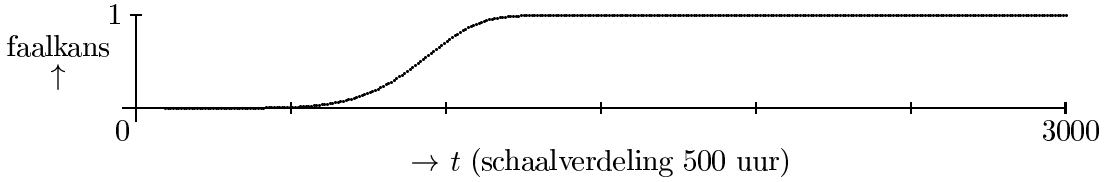
Tabel 4: Optimalisatie van groep 1 (Tabel 1) met IHG *bij* brandkental $b_1 = 1$ en strafkosten $p_1 = 3000$.



Figuur 10: De kostenfunctie naar T bij vaste $\underline{k}$ ($k_1 = 1$) van groep 1 (Tabel 1) *bij* brandkental $b_1 = 1$ en strafkosten $p_1 = 3000$.

groep	#	a	c	p	b	type	α	λ
1	10	100	100	3000	1	Weibull	7.61	7.44E-4
2	10	10	10	3000	1	Weibull	6.82	1.03E-3

Tabel 5: De gegevens van groep 1 en 2.



Figuur 11: De levensduurverdeling van de lampen uit groep 2 (Tabel 5).

brandkental de vermenigingsvuldigingsfactor M_1 beïnvloedt, zie Formule (8). In Figuur 10 is de kostenfunctie weergegeven en in Tabel 4 de optimalisatie resultaten voor deze aanpassing van de strafkosten voor groep 1.

De volgende uitbreiding is het combineren van groep 1 met een andere groep, nl. groep 2. In Tabel 5 zijn beide groepen opgenomen.

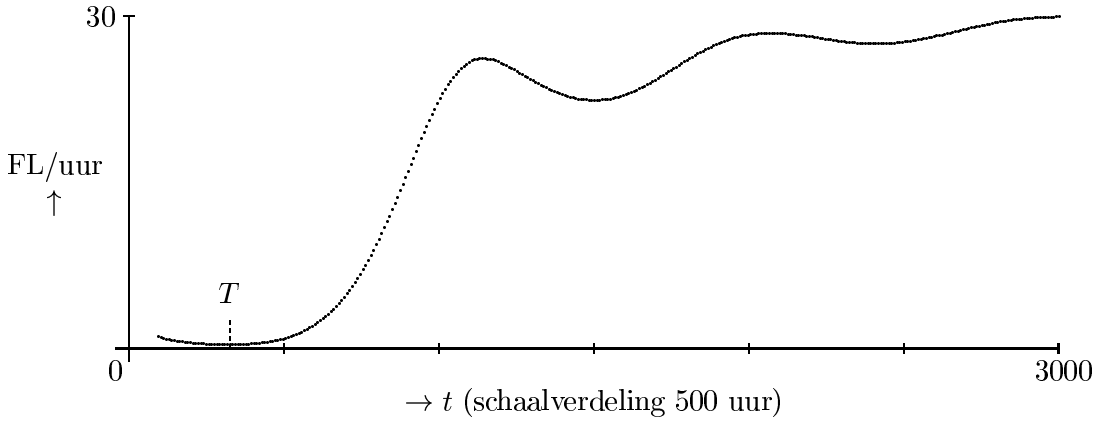
In Figuur 11 is de levensduurverdeling van de lampen uit groep 2 gegeven en in Tabel 6 de resultaten van optimalisatie van groep 2 *alleen*. Deze resultaten zijn duidelijk in overeenstemming met de kostenfunctie zoals afgebeeld in Figuur 12.

Merk op dat de vervangingskosten van groep 2 aanmerkelijk lager zijn dan die voor groep 1 en tevens aanmerkelijk lager zijn dan de basisvervangingskosten A. Als we groep 1 en 2 combineren zijn er twee, voor de hand liggende, combinaties om te onderzoeken: $k_1 = k_2 = 1$; $k_1 = 2$ en $k_2 = 1$. Wat betreft het eerste geval, T zou dan ergens tussen 328 uur en 540 uur terecht komen. Dat is vrij kostbaar, want de strafkosten van groep 2 zijn hoog en het eerder preventief vervangen van groep 1 is duur. De tweede combinatie is gunstiger, want het eerder preventief vervangen van groep 2 is relatief goedkoop. Het blijkt inderdaad dat combinatie 2 door het systeem gekozen wordt, zie Tabel 7 en Figuur 13.

Als laatste draaien we de situatie om: vervangen van groep 2 wordt nu duur en van groep 1 relatief goedkoop. De preventieve vervanging van groep 2 wordt dan toonaangevend: de tijdsduur tussen preventieve vervangingen van groep 1 wordt nu aangepast aan die van groep 2. In Tabel 8 zijn de gewijzigde gegevens voor groep 1 en 2 afgebeeld. Merk op dat $a_1 = c_1 = 10$, voorheen was dit 100, en dat $a_2 = c_2 = 100$, voorheen was dit 10. In tabel Tabel 9 worden de optimalisatie resultaten getoond, en in Figuur 14 de kostenfunctie.

T	328 uur	1.9 weken
k_1	1	
Kosten met straf	*FL.* 0.39/uur	*FL.* 3416/jaar
Kosten	*FL.* 0.34/uur	*FL.* 2978/jaar
Uitval	0.000019 defecten/uur	0.17 defecten/jaar

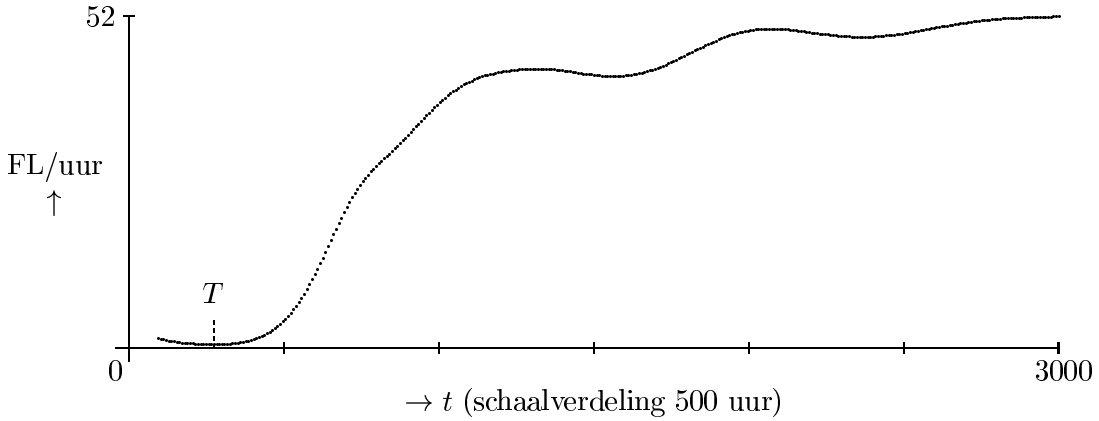
Tabel 6: Optimalisatie van groep 2 (Tabel 5) met IHG.



Figuur 12: De kostenfunctie naar T bij vaste $\underline{k}$ ($k_2 = 1$) van groep 2 (Tabel 5).

T	278 uur	1.6 weken
k_1	2	
k_2	1	
Kosten met straf	*FL.* 0.67/uur	*FL.* 5869/jaar
Kosten	*FL.* 0.58/uur	*FL.* 5081/jaar
Uitval	0.000029 defecten/uur	0.25 defecten/jaar

Tabel 7: Optimalisatie van groep 1 en 2 uit Tabel 5 met IHG.



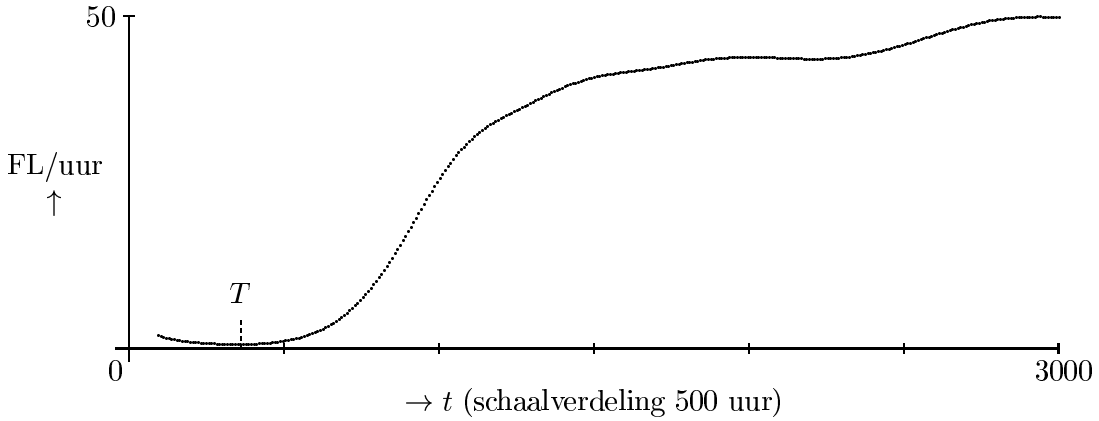
Figuur 13: De kostenfunctie naar T bij vaste $\underline{k}$ ($k_1 = 2$, $k_2 = 1$) van groep 1 en 2 uit Tabel 5.

groep	#	a	c	p	b	type	α	λ
1	10	10	10	3000	1	Weibull	7.61	7.44E-4
2	10	100	100	3000	1	Weibull	6.82	1.03E-3

Tabel 8: De gewijzigde gegevens van groep 1 en 2.

T	358 uur	2.1 weken
k_1	1	
k_2	1	
Kosten met straf	*FL.* 0.69/uur	*FL.* 6044/jaar
Kosten	*FL.* 0.59/uur	*FL.* 5168/jaar
Uitval	0.000032 defecten/uur	0.28 defecten/jaar

Tabel 9: Optimalisatie van groep 1 en 2 uit Tabel 8 met IHG.



Figuur 14: De kostenfunctie naar T bij vaste $\underline{k}$ ($k_1 = 1$, $k_2 = 1$) van groep 1 en 2 uit Tabel 8.

8.4.2 Geval 2

Het hier behandelde voorbeeld gaat over OBR. Er worden twee groepen bekeken waarvan de aanvankelijke gegevens zijn opgenomen in Tabel 10.

groep	#	a	c	p	b	type	α	λ
1	10	100	100	100	1	Weibull	29.3	2.06E-4
2	10	100	100	100	1	Weibull	2.46	4.01E-4

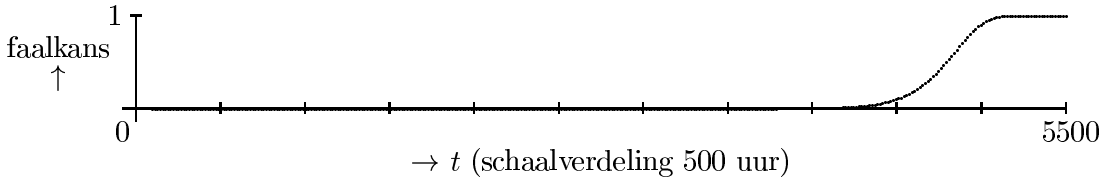
Tabel 10: De gegevens van groep 1 en 2.

Ook in dit voorbeeld wordt gebruik gemaakt van enigszins geconstrueerde levensduurverdelingen. De lampen in groep 1 zijn relatief betrouwbaar, deze branden tot de 4000 uur zonder al te veel problemen. De lampen in groep 2 zijn relatief onbetrouwbaar, na 750 uur begint de uitval al duidelijk vorm aan te nemen. Zie resp. Figuur 15 en 16.

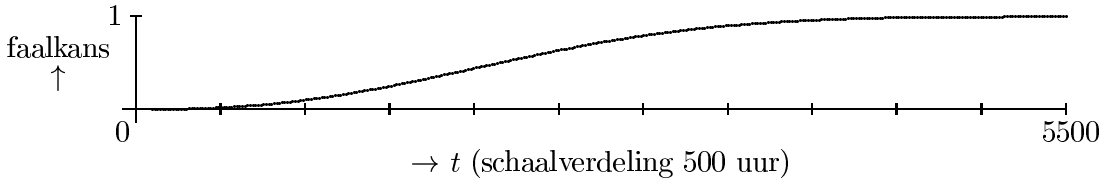
Merk op dat de vervangingskosten van beide groepen vergelijkbaar zijn en de strafkosten van beide groepen niet hoog zijn. Dat zou kunnen betekenen dat de t_1 van groep 1 betrekkelijk hoog is. Immers, een 'hoge' t_1 betekent uitstel van preventief onderhoud. Dat is geen probleem want groep 1 is betrouwbaar en daarbij zijn de strafkosten laag. Voor groep 2 geldt dat de opportunities komende van groep 1 te verwaarlozen zijn. Deze analyse is in overeenstemming met de optimalisatiewaarden die het prototype vindt, deze zijn afgebeeld in Tabel 11.

Indien de vervangingskosten voor groep 1 verlaagd worden, bijvoorbeeld $a_1 = c_1 = 10$, dan is het voor groep 1 zinvoller om van opportunities uit groep 2 gebruik te maken. Kortom, het valt te verwachten dat t_1 lager zal worden om eerder preventieve vervangingen toe te laten. De optimalisatieresultaten bij $a_1 = c_1 = 10$ zijn afgebeeld in Tabel 12, inderdaad is t_1 lager, nl. 1354 uur, dit was eerst 2645 uur.

Verhogen van de strafkosten van groep 1 dringt t_1 nog verder omlaag, zie Tabel 13. Merk op dat het aantal defecten per tijdseenheid in alle drie de tabellen gelijk blijft. Dat komt omdat t_2 ongewijzigd blijft en t_1 weliswaar wijzigt maar groep 2 is dusdanig betrouwbaar op het interval [613,2645] dat dit geen effect heeft.



Figuur 15: Levensduurverdeling van groep 1.



Figuur 16: Levensduurverdeling van groep 2.

t_1	2645	3.7 maanden
t_2	332	2 weken
Kosten met straf	*FL.* 0.38/uur	*FL.* 3329/jaar
Kosten	*FL.* 0.26/uur	*FL.* 2278/jaar
Uitval	0.0012 defecten/uur	3 defecten/jaar

Tabel 11: Optimalisatie van groep 1 en 2 uit Tabel 10 met OBR.

t_1	1354	1.9 maanden
t_2	332	2 weken
Kosten met straf	*FL.* 0.35/uur	*FL.* 3066/jaar
Kosten	*FL.* 0.23/uur	*FL.* 2015/jaar
Uitval	0.0012 defecten/uur	3 defecten/jaar

Tabel 12: Optimalisatie van groep 1 en 2 uit Tabel 10 waarbij $a_1 = c_1 = 10$.

t_1	613	3.6 weken
t_2	332	2 weken
Kosten met straf	*FL.* 0.36/uur	*FL.* 3154/jaar
Kosten	*FL.* 0.23/uur	*FL.* 2015/jaar
Uitval	0.0012 defecten/uur	3 defecten/jaar

Tabel 13: Optimalisatie van groep 1 en 2 uit Tabel 10 waarbij $a_1 = c_1 = 10$ en $p_1 = 1000$.

8.4.3 Geval 3

In deze sectie wordt een verkeersplein behandeld gebaseerd op een fasediagram dat ons door Nederland Haarlem ter hand is gesteld (Amsteldijk-Vrijheidslaan). Op basis van bovengenoemde realistische kosten en lampeigenschappen zullen een aantal groepsindelingen en groepskosten vastgesteld worden waarna voor zowel IHG als OBR een optimalisatie is uitgevoerd met het prototype. De gegevens over kosten uit de praktijk zijn de volgende.

1. Voor elke remplace-actie, preventief of correctief, en ongeacht de duur van de actie, wordt *Fl.* 250,– gerekend.
2. De huur van een actiewagen, een aanhanger met benodigdheden voor o.a. de wegafzetting, kost *FL.* 90,– per dag.
3. Per lichtpunt tot een hoogte van 3 meter wordt *FL.* 4,– gerekend, dit is exclusief kosten van de lamp.
4. Per lichtpunt van 3 meter en hoger wordt *Fl.* 19,– gerekend, wederom exclusief de kosten van de lamp.
5. Eén lamp 220V–100W kost *FL.* 2,64.
6. Eén lamp 220V–75W kost *FL.* 2,53.
7. Eén lamp 40V–40W kost *FL.* 8,53.
8. Eén lamp 10V–35W kost *FL.* 14,74.

De locatie van lamp tot een hoogte van 3 meter en vanaf 3 meter en hoger duiden we in het vervolg aan met 'laag' resp. 'hoog'.

Philips heeft gegevens over de levensduurverdelingen van enkele typen lampen verstrekt. Philips hanteert voor deze typen lampen een Weibull-verdeling. De gegevens van Philips bestaan uit enkele grafieken die een verband geven tussen de brandduur bij constant branden en faalkans. In elk van deze grafieken zijn twee punten aangegeven door Philips, het punt waar geldt $F(t) = 2\%$ en het punt waar geldt $F(t) = 50\%$.

Type	brandduur 1	faalkans 1	brandduur 2	faalkans 2
Halogeen lamp 10V	2800 uur	2%	8600 uur	50%
Gloeilamp 230V	3000 uur	2%	8000 uur	50%
Laag-voltage lamp 40V	4400 uur	2%	8000 uur	50%

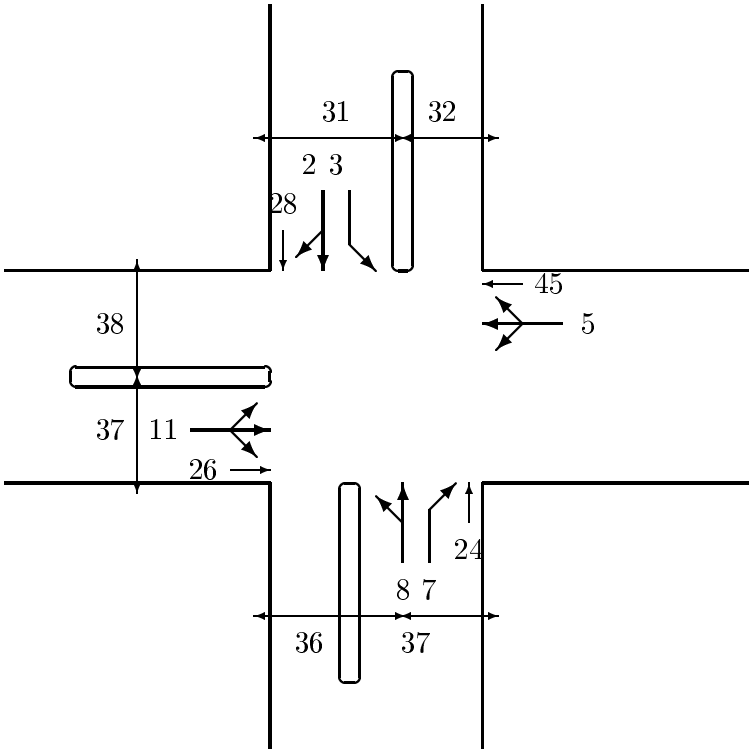
Uit deze gegevens kan, gezien het feit dat het punten uit een bekende levensduurverdeling zijn nl. de Weibull-verdeling, per lamptype de vormparameter α en de schaalparameter λ worden afgeleid. Hieronder zal een voorbeeld gegeven worden van een dergelijke afleiding. Bij het gebruik van een BOS in de praktijk verdient het echter aanbeveling de parameters van de levensduurverdeling rechtstreeks van de fabrikant te betrekken.

Laten b_1 en b_2 twee brandduren zijn en laten f_1 en f_2 de bijbehorende faalkansen zijn op een Weibull-verdeling. Opgelost moeten nu α en λ uit de volgende twee vergelijkingen (zie de specificatie van de Weibull-verdeling in Paragraaf 3).

$$f_1 = 1 - e^{-(\lambda b_1)^\alpha} \quad f_2 = 1 - e^{-(\lambda b_2)^\alpha}$$

De oplossing is wordt gegeven door de volgende uitdrukkingen.

$$\alpha = \frac{\ln\left(\frac{\ln(1-f_1)}{\ln(1-f_2)}\right)}{\ln(\frac{b_1}{b_2})} = \frac{\ln\left(\frac{\ln(1-f_2)}{\ln(1-f_1)}\right)}{\ln(\frac{b_2}{b_1})}$$



Figuur 17: Amsteldijk–Vrijheidslaan.

$$\lambda = \frac{[-\ln(1-f_1)]^{\frac{1}{\alpha}}}{b_1} = \frac{[-\ln(1-f_2)]^{\frac{1}{\alpha}}}{b_2}$$

Gebruik makend van deze betrekkingen kan men de volgende parameters afleiden.

Type	vormparameter α	schaalparameter λ
Halogeen lamp 10V	3.151	1.035E-4
Gloeilamp 230V	3.605	1.129E-4
Laag-voltage lamp 40V	5.914	1.175E-4

We nemen aan dat een remplace-actie, preventief of correctief, niet langer kost dan een halve dag. De basistariefkosten A komen zo op $FL.\ 250,-+45,-=295,-$. Tevens relateren we de eerder genoemde lampkosten en lampen. We doen dit t.b.v. het voorbeeld en op intuïtieve gronden, de gegevens komen immers uit verschillende bronnen. Resumerend gebruiken we de volgende gegevens over lampen.

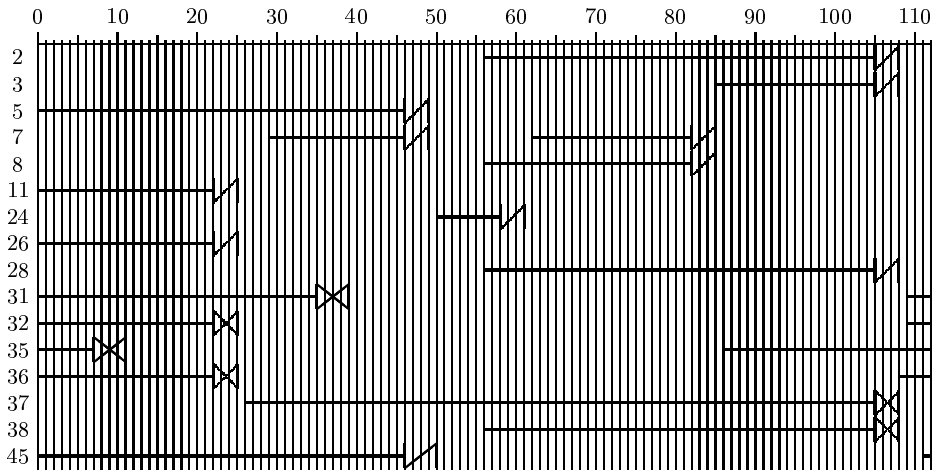
Type	vormparameter α	schaalparameter λ	prijs
Halogeen lamp 10V	3.151	1.035E-4	$FL.\ 14,74$
Gloeilamp 230V	3.605	1.129E-4	$FL.\ 2,64$
Laag-voltage lamp 40V	5.914	1.175E-4	$FL.\ 8,53$

Het voorbeeld Amsteldijk–Vrijheidslaan is enigszins vereenvoudigd omdat met name het aantal fasen te groot zou worden om hier goed te kunnen behandelen. De variant die hier behandeld wordt is in de Figuren 17 en 18, en in Tabel 14 beschreven.

Op basis van de gegevens in Tabel 14 construeren we een aantal groepsindelingen. De eerste groepsindeling is op lamptype: één groep met alle laag-voltage 40V lampen en één groep met alle gloeilampen 230V. De eerste worden in het vervolg met 40V en de tweede met 230V aangeduid. Merk op dat hierdoor beide groepen lampen bevatten met sterk uiteenlopende brandkentallen. De

fase	locatie	kleur	aant. lampen	brandkental b	lamptype
2	hoog	rood	1	$60/112 = 0.536$	Laag-voltage lamp 40V
		geel	1	$3/112 = 0.027$	Laag-voltage lamp 40V
		groen	1	$49/112 = 0.438$	Laag-voltage lamp 40V
3	laag	rood	1	$89/112 = 0.795$	Laag-voltage lamp 40V
		geel	1	$3/112 = 0.027$	Laag-voltage lamp 40V
		groen	1	$20/112 = 0.179$	Laag-voltage lamp 40V
5	hoog	rood	1	$63/112 = 0.563$	Laag-voltage lamp 40V
		geel	1	$3/112 = 0.027$	Laag-voltage lamp 40V
		groen	1	$46/112 = 0.411$	Laag-voltage lamp 40V
7	laag	rood	1	$69/112 = 0.616$	Laag-voltage lamp 40V
		geel	1	$6/112 = 0.054$	Laag-voltage lamp 40V
		groen	1	$37/112 = 0.330$	Laag-voltage lamp 40V
8	hoog	rood	1	$83/112 = 0.741$	Laag-voltage lamp 40V
		geel	1	$3/112 = 0.027$	Laag-voltage lamp 40V
		groen	1	$26/112 = 0.232$	Laag-voltage lamp 40V
11	hoog	rood	1	$87/112 = 0.777$	Laag-voltage lamp 40V
		geel	1	$3/112 = 0.027$	Laag-voltage lamp 40V
		groen	1	$22/112 = 0.196$	Laag-voltage lamp 40V
24	laag	rood	1	$101/112 = 0.902$	Gloeilamp 230V
		geel	1	$3/112 = 0.027$	Gloeilamp 230V
		groen	1	$8/112 = 0.071$	Gloeilamp 230V
26	laag	rood	1	$87/112 = 0.777$	Gloeilamp 230V
		geel	1	$3/112 = 0.027$	Gloeilamp 230V
		groen	1	$22/112 = 0.196$	Gloeilamp 230V
28	laag	rood	1	$60/112 = 0.536$	Gloeilamp 230V
		geel	1	$3/112 = 0.027$	Gloeilamp 230V
		groen	1	$49/112 = 0.438$	Gloeilamp 230V
31	laag	rood	2	$70/112 = 0.625$	Gloeilamp 230V
		groen	2	$40/112 = 0.357$	Gloeilamp 230V
32	laag	rood	2	$84/112 = 0.750$	Gloeilamp 230V
		groen	2	$26.5/112 = 0.237$	Gloeilamp 230V
35	laag	rood	2	$75/112 = 0.670$	Gloeilamp 230V
		groen	2	$35/112 = 0.313$	Gloeilamp 230V
36	laag	rood	2	$83/112 = 0.741$	Gloeilamp 230V
		groen	2	$27.5/112 = 0.246$	Gloeilamp 230V
37	laag	rood	2	$30/112 = 0.268$	Gloeilamp 230V
		groen	2	$80.5/112 = 0.719$	Gloeilamp 230V
38	laag	rood	2	$60/112 = 0.536$	Gloeilamp 230V
		groen	2	$50.5/112 = 0.451$	Gloeilamp 230V
45	laag	rood	1	$61/112 = 0.545$	Gloeilamp 230V
		geel	1	$4/112 = 0.036$	Gloeilamp 230V
		groen	1	$47/112 = 0.420$	Gloeilamp 230V

Tabel 14: De configuratie gegevens voor de VLI Amsteldijk–Vrijheidslaan.



Figuur 18: Fasediagram Amstelkade–Vrijheidslaan. Een doorgetrokken streep is 'groen', een schuine lijn is 'geel', een kruis is 'knipper groen', waarbij een 1 op 1 verhouding tussen 'aan' en 'uit' wordt ondersteld.

groep	#	a	c	p	b	type	α	λ
1 (40V)	18	405.54	405.54	1000	0.795	Weibull	5.914	1.175E-4
2 (230V)	36	239.04	239.04	1000	0.902	Weibull	3.605	1.129E-4

Tabel 15: Indeling in twee groepen op lamptype.

groepsindeling is samengevat in Tabel 15. We lopen de gegevens even langs. De kosten a_j en c_j zijn hetzelfde genomen, dat ligt hier voor de hand want in beide gevallen gaat het om kosten voor het vervangen van alle lampen uit groep j zonder basistarief of straf in aanmerking te nemen. Het is goed denkbaar dat in de praktijk deze kosten verschillen omdat er specifieke kosten worden bijgeteld of in mindering gebracht. De preventieve vervangingskosten a_1 zijn als volgt bepaald: $12 \times (FL.\ 19,- + 8.53) + 6 \times (FL.\ 4,- + 8.53) = FL.\ 405.54$. Immers, er hangen 12 lampen 'hoog' en 6 'laag'. De andere kosten zijn op een soortgelijke manier berekend. De strafkosten zijn laag gehouden zodat later deze kosten opgevoerd kunnen worden als de betrouwbaarheid te laag is, maar de strafkosten zijn wel hoger dan de andere kosten gekozen in verband met de voorwaarden van de lemma's uit eerdere paragrafen. Uit veiligheidsoverwegingen is de lamp met het hoogste brandkental maatgevend voor de groep.

De resultaten van optimalisatie met IHG staan in Tabel 16.
De verwachte uitval is in dit schema 0.000086 lampen per uur, per jaar geeft dat een uitval van $0.000086 \times 24 \times 365$ is 0.75 lampen per jaar. Dat lijkt een redelijk ijkpunt om bij verdere verfijningen van de indeling naar te streven.

T	2127 uur	2.9 maanden
k_1	2	
k_2	1	
Kosten met straf	$FL.$ 0.46/uur	$FL.$ 4029/jaar
Kosten	$FL.$ 0.37/uur	$FL.$ 3241/jaar
Uitval	0.000086 defecten/uur	0.75 defecten/jaar

Tabel 16: Resultaat van optimalisatie met IHG van de groepsgegevens uit Tabel 15.

t_1	2076 uur	2.9 maanden
t_2	552 uur	3.3 weken
Kosten met straf	*FL.* 0.63/uur	*FL.* 5519/jaar
Kosten	*FL.* 0.28/uur	*FL.* 2453/jaar
Uitval	0.000352 defecten/uur	3.1 defecten/jaar

Tabel 17: Resultaat van optimalisatie met OBR van de groepsgegevens uit Tabel 15.

groep	#	a	c	p	b	type	α	λ
1 (Rood, 40V)	6	135.18	135.18	200	0.795	Weibull	5.914	1.175E-4
2 (Geel, 40V)	6	135.18	135.18	200	0.054	Weibull	5.914	1.175E-4
3 (Groen, 40V)	6	135.18	135.18	200	0.438	Weibull	5.914	1.175E-4
4 (Rood, 230V)	12	79.68	79.68	200	0.902	Weibull	3.605	1.129E-4
5 (Geel, 230V)	12	79.68	79.68	200	0.036	Weibull	3.605	1.129E-4
6 (Groen, 230V)	12	79.68	79.68	200	0.719	Weibull	3.605	1.129E-4

Tabel 18: Indeling in 6 groepen op lamptype en 'kleur'.

De resulaten van optimalisatie met OBR worden gegeven in Tabel 17.
Het OBR schema geeft dus, naar verwachting, 4 $\times$ zoveel uitval: 3.1 defecten/jaar tegen 0.75 defecten/jaar. De kosten zonder strafkosten zijn wel lager: *FL.* 2453/jaar tegen *FL.* 3241/jaar. De betrouwbaarheid valt echter niet te verbeteren, althans, bij veel hogere strafkosten treedt slechts een zeer kleine verbetering in de betrouwbaarheid op. Ook in de nog te behandelen indelingen scoort OBR slechter op betrouwbaarheid, terwijl IHG bij vergelijkbare betrouwbaarheid beter scoort qua kosten. Er wordt daarom van af gezien OBR resultaten te tonen en te behandelen voor het geval Amstelkade–Vrijheidslaan.
We gaan de groepsindeling van Tabel 15 nu verfijnen door groep 1 en 2 elk te splitsen in drie groepen op 'kleur'. Het resultaat van de opsplitsing is afgebeeld in Tabel 18.

De kosten a en c zijn bepaald op dezelfde wijze als ten behoeve van Tabel 15. Omdat de kosten a en c lager zijn geworden, kunnen de initiële strafkosten ook omlaag. Bij de brandkentallen is de hoogste uit de groep weer maatgevend voor de groep; we zullen in de rest van deze paragraaf altijd het hoogste brandkental maatgevend maken. Optimalisatie met IHG resulteert nu in het schema dat getoond wordt in Tabel 19.
Deze groepsindeling geeft een verbetering te zien in de kosten t.o.v. de groepsindeling in Tabel 15, nl. 14 cent per uur, een verbetering van bijna 38%. De betrouwbaarheid is echter lager, 0.000115 tegen

T	2960 uur	4.1 maanden
k_1	2	
k_2	24	
k_3	3	
k_4	1	
k_5	19	
k_6	1	
Kosten met straf	*FL.* 0.25/uur	*FL.* 2190/jaar
Kosten	*FL.* 0.23/uur	*FL.* 2014/jaar
Uitval	0.000115 defecten/uur	1.01 defecten/jaar

Tabel 19: iResultaat van optimalisatie met IHG van de groepsgegevens uit Tabel 18.

T	2673 uur	3.7 maanden
k_1	2	
k_2	25	
k_3	3	
k_4	1	
k_5	19	
k_6	1	
Kosten met straf	*FL.* 0.28/uur	*FL.* 2453/jaar
Kosten	*FL.* 0.24/uur	*FL.* 2102/jaar
Uitval	0.000083 defecten/uur	0.73 defecten/jaar

Tabel 20: Resultaat van optimalisatie met IHG van de groepsgegevens uit Tabel 18 met verhoging van de strafkosten naar *FL.* 500,–.

groep	#	a	c	p	b	type	α	λ
1 (Rood, hoog, 40V)	4	110.12	110.12	500	0.777	Weibull	5.914	1.175E-4
2 (Geel, hoog, 40V)	4	110.12	110.12	500	0.027	Weibull	5.914	1.175E-4
3 (Groen, hoog, 40V)	4	110.12	110.12	500	0.438	Weibull	5.914	1.175E-4
4 (Rood, laag, 40V)	2	25.06	25.06	500	0.795	Weibull	5.914	1.175E-4
5 (Geel, laag, 40V)	2	25.06	25.06	500	0.054	Weibull	5.914	1.175E-4
6 (Groen, laag, 40V)	2	25.06	25.06	500	0.330	Weibull	5.914	1.175E-4
7 (Rood, 230V)	12	79.68	79.68	500	0.902	Weibull	3.605	1.129E-4
8 (Geel, 230V)	12	79.68	79.68	500	0.036	Weibull	3.605	1.129E-4
9 (Groen, 230V)	12	79.68	79.68	500	0.719	Weibull	3.605	1.129E-4

Tabel 21: Indeling in 9 groepen op lamptype, locatie en 'kleur'.

0.000086 defecten per uur. Door de strafkosten te verhogen van *FL.* 200,– naar *FL.* 500,– kunnen we de betrouwbaarheid verbeteren tot een vergelijkbaar niveau, zie Tabel 20.
Merk echter op dat de groepen wat betreft brandkental heterogeen zijn: als het model klopt, iets dat door vergelijking met de praktijk moet worden onderzocht, zou de betrouwbaarheid in de praktijk hoger kunnen uitvallen.

We kunnen proberen de resultaten nog verder te verbeteren door verder te verfijnen. Zo zou het brandkental een grotere rol in de groepsindeling kunnen spelen. Tevens zouden we in aanmerking kunnen nemen dat er lampen 'hoog' en 'laag' zitten. Als sluitstuk bekijken we de groepsindeling waarin 'elke lamp' als groep opgevat is.

In tabel 21 is een groepsindeling gegeven waarin onderscheid is gemaakt tussen 'hoge' en 'lage' lampen. Na een paar keer proberen met de strafkosten blijkt wederom een straf van *FL.* 500,– per groep een betrouwbaarheid te geven die maar weinig afwijkt van ons ijkpunt van 0.000086 defecten per uur. De kosten per tijdseenheid wijken nauwelijks af van de indeling is zes groepen uit Tabel 18. Het resultaat, zie Tabel 22, lijkt dus vergelijkbaar, maar in de praktijk zal dit mogelijk geen werkbaar schema zijn. Bijvoorbeeld, de hoge, gele lampen worden slechts eens in 15.6 jaar vervangen. Op grond van dergelijke meta-overwegingen moeten we waarschijnlijk concluderen dat dit schema geen verbetering is.
We kunnen nu nog proberen een verbetering te realiseren door meer waarde aan de brandkentallen te hechten en tevens de extremiteiten als $k_2 = 49$ weg te werken. We voegen daartoe in de categorie van de 40V lampen de gele lampen bij de groene. De lampen in de categorie 230V verdelen we in drie groepen: brandkental groter gelijk 70/112, brandkental tussen 30/112 en 70/112 en brandkental kleiner gelijk 30/112. In Tabel 23 is de groepsindeling gegeven. Er is geen speciale reden voor de keuze van deze intervallen, de indeling is op slechts op intuïtieve gronden zo gekozen.

T	2798 uur	3.9 maanden
k_1	2	
k_2	49	
k_3	3	
k_4	1	
k_5	21	
k_6	3	
k_7	1	
k_8	18	
k_9	1	
Kosten met straf	*FL.* 0.28/uur	*FL.* 2453/jaar
Kosten	*FL.* 0.24/uur	*FL.* 2102/jaar
Uitval	0.000085 defecten/uur	0.74 defecten/jaar

Tabel 22: Resultaat van optimalisatie met IHG van de gegevens uit Tabel 21.

groep	#	a	c	p	b	type	α	λ
1 (Rood, hoog, 40V)	4	110.12	110.12	550	0.777	Weibull	5.914	1.175E-4
2 (Geel & Groen, hoog, 40V)	8	220.24	220.24	550	0.438	Weibull	5.914	1.175E-4
3 (Rood, laag, 40V)	2	25.06	25.06	550	0.795	Weibull	5.914	1.175E-4
4 (Geel & Groen, laag, 40V)	4	50.12	50.12	550	0.330	Weibull	5.914	1.175E-4
5 ($b_i \geq 70/112$, 230V)	12	79.68	79.68	550	0.902	Weibull	3.605	1.129E-4
6 ($30/112 < b_i < 70/112$, 230V)	14	92.96	92.96	550	0.545	Weibull	3.605	1.129E-4
7 ($b_i \leq 30/112$, 230V)	10	66.40	66.40	550	0.246	Weibull	3.605	1.129E-4

Tabel 23: Indeling in 7 groepen op lamptype, locatie en brandkental.

T	2832 uur	3.9 maanden
k_1	2	
k_2	3	
k_3	1	
k_4	3	
k_5	1	
k_6	1	
k_7	3	
Kosten met straf	*FL.* 0.31/uur	*FL.* 2716/jaar
Kosten	*FL.* 0.26/uur	*FL.* 2278/jaar
Uitval	0.000085 defecten/uur	0.74 defecten/jaar

Tabel 24: Resultaat van optimalisatie met IHG van de gegevens uit Tabel 23.

Deze groepsindeling levert geen kostenverbetering, maar het schema is wel fraaier. Dat deze groepsindeling groepen met combinaties van lampenkleuren kent maakt de indeling nog minder aantrekkelijk; de uitvoering van het vervangingsschema zal mogelijk fouten in de hand werken. Overigens is dat maar een veronderstelling die in de praktijk zou moeten worden getoetst. Daarbij, als 'complexe' schema's veel kostenwinst leveren moet misschien onderzocht worden of er geen technische middelen zijn om de uitvoering te verbeteren.
Als laatste bekijken we de situatie waarin alle lampen als groep worden opgevat, behalve de lampen die evident in dezelfde signaalgroep zitten, i.e., de voetgangerslichten, zie Tabel 25. Deze aanpak levert een verlaging van de kosten.
Het resultaat van optimalisatie is gegeven in Tabel 26. We kunnen dit resultaat comprimeren door die groepen met dezelfde waarde voor 'k' bij elkaar te plaatsen, zie Tabel 27. Dit levert een indeling in 11 collecties die men zou kunnen opvatten als een indeling is 11 groepen. Het optimalisatieproces heeft tot bepaalde combinaties geleid. Dit roept de vraag op of het altijd verstandig is om eerst een optimalisatie van een groepsindeling op een zo fijn mogelijk niveau uit te voeren, de structuur van het gegenereerde schema te onderzoeken en zo tot een afgewogen groepsindeling te komen. De vraag of dit een bruikbare methode is kan op dit moment niet beantwoord worden.
Nadeel van het bovenstaande schema is gelegen in de extreem lange cycli, wel tot 18 jaar, en de complexiteit. Het lijkt erop dat de groepsindeling in Tabel 18 het schema oplevert dat economisch een redelijk resultaat geeft en dat ook redelijk uitvoerbaar is. Echter, ook dit schema lijdt onder vervangingscycli die erg lang zijn, nl. voor de gele lampen die op grond van meta-overwegingen niet aanvaardbaar zijn. Men zou deze cycli kunnen bekorten door combinaties met groen. We zullen dit hier niet doen.

8.5 Enkele uitzonderingsgevallen

In deze Paragraaf worden kort enkele situaties bekeken die zich mogelijk kunnen voordoen in de praktijk en wier behandeling mogelijk niet evident is.

8.5.1 Lampen

Dubbel uitgevoerde lampen kunnen, naar ons idee vooralsnog, het best als twee aparte lampen worden opgevat. Echter, men kan hier ook een andere strategie op baseren: wacht met preventief onderhoud totdat er 'genoeg' eerste lampen stuk zijn. Dit is een nieuw probleem waar nu niet veel over te zeggen valt.

Lampen worden soms aangestuurd met een lagere spanning dan gebruikelijk is voor het type lamp. Dit is geen probleem indien een levensduurverdeling van de lamp bij gebruik onder deze lagere spanning

groep	#	a	c	p	b	type	α	λ
1	1	27.53	27.53	350	0.536	Weibull	5.914	1.175E-4
2	1	27.53	27.53	350	0.027	Weibull	5.914	1.175E-4
3	1	27.53	27.53	350	0.438	Weibull	5.914	1.175E-4
4	1	12.53	12.53	350	0.795	Weibull	5.914	1.175E-4
5	1	12.53	12.53	350	0.027	Weibull	5.914	1.175E-4
6	1	12.53	12.53	350	0.179	Weibull	5.914	1.175E-4
7	1	27.53	27.53	350	0.563	Weibull	5.914	1.175E-4
8	1	27.53	27.53	350	0.027	Weibull	5.914	1.175E-4
9	1	27.53	27.53	350	0.411	Weibull	5.914	1.175E-4
10	1	12.53	12.53	350	0.616	Weibull	5.914	1.175E-4
11	1	12.53	12.53	350	0.054	Weibull	5.914	1.175E-4
12	1	12.53	12.53	350	0.330	Weibull	5.914	1.175E-4
13	1	12.53	12.53	350	0.741	Weibull	5.914	1.175E-4
14	1	12.53	12.53	350	0.027	Weibull	5.914	1.175E-4
15	1	12.53	12.53	350	0.232	Weibull	5.914	1.175E-4
16	1	27.53	27.53	350	0.777	Weibull	5.914	1.175E-4
17	1	27.53	27.53	350	0.027	Weibull	5.914	1.175E-4
18	1	27.53	27.53	350	0.196	Weibull	5.914	1.175E-4
19	1	6.64	6.64	350	0.902	Weibull	3.605	1.129E-4
20	1	6.64	6.64	350	0.027	Weibull	3.605	1.129E-4
21	1	6.64	6.64	350	0.071	Weibull	3.605	1.129E-4
22	1	6.64	6.64	350	0.777	Weibull	3.605	1.129E-4
23	1	6.64	6.64	350	0.027	Weibull	3.605	1.129E-4
24	1	6.64	6.64	350	0.196	Weibull	3.605	1.129E-4
25	1	6.64	6.64	350	0.536	Weibull	3.605	1.129E-4
26	1	6.64	6.64	350	0.027	Weibull	3.605	1.129E-4
27	1	6.64	6.64	350	0.438	Weibull	3.605	1.129E-4
28	2	13.28	13.28	350	0.625	Weibull	3.605	1.129E-4
29	2	13.28	13.28	350	0.357	Weibull	3.605	1.129E-4
30	2	13.28	13.28	350	0.750	Weibull	3.605	1.129E-4
31	2	13.28	13.28	350	0.237	Weibull	3.605	1.129E-4
32	2	13.28	13.28	350	0.670	Weibull	3.605	1.129E-4
33	2	13.28	13.28	350	0.313	Weibull	3.605	1.129E-4
34	2	13.28	13.28	350	0.741	Weibull	3.605	1.129E-4
35	2	13.28	13.28	350	0.246	Weibull	3.605	1.129E-4
36	2	13.28	13.28	350	0.268	Weibull	3.605	1.129E-4
37	2	13.28	13.28	350	0.719	Weibull	3.605	1.129E-4
38	2	13.28	13.28	350	0.536	Weibull	3.605	1.129E-4
39	2	13.28	13.28	350	0.451	Weibull	3.605	1.129E-4
40	1	6.64	6.64	350	0.545	Weibull	3.605	1.129E-4
41	1	6.64	6.64	350	0.036	Weibull	3.605	1.129E-4
42	1	6.64	6.64	350	0.420	Weibull	3.605	1.129E-4

Tabel 25: Indeling in 42 groepen.

T	3989 uur	5.5 maanden
k_1	2	
k_2	36	
k_3	2	
k_4	1	
k_5	31	
k_6	5	
k_7	2	
k_8	36	
k_9	2	
k_{10}	1	
k_{11}	16	
k_{12}	3	
k_{13}	1	
k_{14}	31	
k_{15}	4	
k_{16}	1	
k_{17}	36	
k_{18}	5	
k_{19}	1	
k_{20}	18	
k_{21}	7	
k_{22}	1	
k_{23}	18	
k_{24}	2	
k_{25}	1	
k_{26}	18	
k_{27}	1	
k_{28}	1	
k_{29}	1	
k_{30}	1	
k_{31}	2	
k_{32}	1	
k_{33}	2	
k_{34}	1	
k_{35}	2	
k_{36}	2	
k_{37}	1	
k_{38}	1	
k_{39}	1	
k_{40}	1	
k_{41}	13	
k_{42}	1	
Kosten met straf	*FL.* 0.21/uur	*FL.* 1840/jaar
Kosten	*FL.* 0.18/uur	*FL.* 1577/jaar
Uitval	0.000082 defecten/uur	0.72 defecten/jaar

Tabel 26: Elke ‘lamp’ een groep.

groep	k
1,3,7,9,24,31,33,35,36	2
2,8,17	36
4,10,13,16,19,22,25,27,28,29,30,32,34,37,38,39,40,42	1
5,14	31
6,18	5
11	16
12	3
15	4
20,23,26	18
21	7
41	13
T	3989 uur
Kosten met straf	*FL.* 0.21/uur
Kosten	*FL.* 0.18/uur
Uitval	0.000082 defecten/uur

Tabel 27: Een verkorte presentatie van de gegevens uit Tabel 26.

bekend is. Bij lagere spanning is naar ons weten wederom sprake van een verdeling in dezelfde klasse. Vooralsnog ondersteund de software alleen Weibull of Erlang, indien een andere verdeling dan één van deze twee nodig mocht blijken te zijn kan deze eenvoudig worden toegevoegd aan de software.

8.5.2 Afwijkingen van het gegenereerde schema

Een vervangingsschema zoals gegenereerd door een beslissingsondersteunend systeem geeft slechts een advies. Afwijkingen zullen altijd moeten worden toegestaan. Dit beïnvloedt de toepassing van de strategie in de toekomst niet mits de afwijkingen niet te groot zijn. Wat groot is is moeilijk aan te geven in algemene zin. Maar in het volgende doorsnee geval lijkt er duidelijk geen probleem. Stel een remplaceschema heeft bewezen minder uitval per tijdseenheid dan de wegbeheerder eist, verder is $T = 12 weken$ en moet groep 3 met $k_3 = 2$ volgens schema preventief vervangen worden op zaterdag 1 juli 1995. Verplaatsen naar maandag 3 juli levert dan waarschijnlijk geen problemen. Het is echter niet onderzocht welke (financiële) consequenties precies verbonden zijn aan het eenmalig afwijken van een voorgeschreven remplace onder IHG of OBR. Het is dan ook niet mogelijk om nu aan te geven hoe een BOS kan aangeven welk risico een afwijking met zich meebrengt.

8.5.3 Wijzigingen aan de configuratie van de VLI

Het is goed denkbaar dat de configuratie van een VLI wijzigt doordat lantaarns worden toegevoegd of verwijderd, de kosten wijzigen, of de lampen door een andere type vervangen worden etc. In alle gevallen lijkt een route te zijn de toekomstige gewijzigde situatie te analyseren met het BOS en remplaceschema's te vergelijken met het lopende schema. Eventueel zal een nieuwe start gemaakt moeten worden door alle lampen op het plein te vervangen en met een nieuw schema te beginnen.

8.5.4 Oscillatie van OBR

Het kan voorkomen dat bij optimalisatie naar de t_j's of de π_j's het algoritme gaat oscilleren en derhalve niet termineert. Dit valt te bestrijden met dempingsfactoren die in Formule 21 in algemene zin worden beschreven.

9 Aggregatiestrategieën, een aanzet

Hier zij nogmaals opgemerkt dat een belangrijk voordeel van IHG boven OBR is dat de eerste strategie leidt tot vervangingsschema's die in beginsel lang van te voren bekend zijn, terwijl onder OBR pas een preventieve vervanging plaatsvindt als er correctief moet worden ingegrepen. Het voordeel van OBR is echter dat munt kan worden geslagen uit (onvermijdelijke) correctieve vervangingen door er preventieve aan te koppelen.

Onder IHG is het wellicht mogelijk ook meerdere kruispunten in hun onderlinge samenhang te bekijken. Stel dat onder toepassing van IHG een vervangingsplan is opgesteld voor de groepen van een (groot) aantal bij elkaar in de buurt gelegen kruispunten. Het kan nu overweging verdienen om een aantal dicht bij elkaar (in tijd en plaats) gelegen vervangingen te combineren tot een 'werkorder' voor 1 dag. Als bijvoorbeeld een vervanging op kruispunt A is gepland op 2 juni, op kruispunt B op 8 juni en op kruispunt C op 5 juli, dan kan het aanbeveling verdienen deze drie vervangingsklussen op 1 dag uit te voeren. De keuze van het 'beste' tijdstip voor de gezamenlijke remplace-order, dient te zijn gebaseerd op 'boetekosten' (niet te verwarren met de eerder ingevoerde strafkosten voor correctief onderhoud) voor het bewust afwijken van de geplande datum. Het bepalen van deze boetekosten in een IHG strategie is nog niet direct duidelijk en vereist nadere studie, zie ook Paragraaf 8.5.2. In de literatuur zijn wel enige resultaten over aggregatie bekend. Het is echter nog niet duidelijk of deze resultaten gebruikt kunnen worden in de onderhavige context [DSL92].

10 Conclusies, aanbevelingen en vervolgstudies

Dit verslag bestaat uit twee delen. In Deel I van dit verslag zijn twee strategieën beschreven die gevolgd zouden kunnen worden bij het plannen en uitvoeren van de lampremplace van een verkeerslichtinstallatie (VLI). Deze strategieën zijn de Indirecte Hoofdgroepering Strategie (IHG) en de 'Opportunity Based Replacement' Strategie (OBR). Voor beide strategieën zijn wiskundige modellen beschreven en zijn handelingsvoorschriften beschreven om gegeven een VLI de optimale waarden voor de beslissingsvariabelen numeriek te bepalen. Daarbij is in Deel I een inleiding gegeven tot het besliskundige begrippen kader, zijn een drietal met IHG en OBR verwante strategieën kort beschreven en is kort ingegaan op aggregatiestrategieën.

In Deel II van dit verslag is een executeerbare algebraïsche specificatie gegeven van een beslissingsondersteunend systeem (BOS) voor lampremplace gebaseerd op de strategieën IHG en OBR. Deze specificatie is een aanzet voor een volledige specificatie, er zijn immers nog te veel open vragen over de specifieke functionaliteit die een dergelijk systeem moet bezitten om goed te functioneren in de praktijk. De specificatie is executeerbaar in die zin dat de globale werking van en de interactie met het BOS gesimuleerd kan worden, optimalisatierekenwerk is niet mogelijk, daarvoor dient het prototype.

Er is een prototype gebouwd, in Pascal, dat gebaseerd is op de optimalisatie handelingsvoorschriften voor IHG en OBR zoals beschreven in Deel I en de specificatie in Deel II. Het prototype richt zich zuiver en alleen op het genereren van een schema gegeven een VLI configuratie. Het prototype is zijdelings besproken, de code is niet opgenomen in dit verslag.

Onze voorzichtige conclusie is dat met dit onderzoek een redelijk uitgangspunt lijkt te zijn gecreëerd voor de bouw van een eenvoudig BOS voor lampremplace. Het nut in beheersmatige of economische zin van een dergelijk systeem zal echter moeten blijken uit praktijktests. Tevens is met dit onderzoek de basis gelegd voor verder onderzoek van de besliskunde- en informatica-aspecten van lampremplace.

Wij voorzien twee lijnen waarlangs het werk aan een BOS voor lampremplace kan worden voortgezet. De eerste lijn behelst met name het gebruik en functioneren van het systeem in de praktijk. De volgende vragen zou men beantwoord willen zien.

- Zijn de wiskundige modellen voor IHG en OBR adequaat? Dat wil zeggen, ontwikkelen de kosten en de defecten zich zoals voorspeld? Om deze testen goed uit te voeren zal een statistisch verantwoord testplan ontworpen moeten worden, zodat eventuele afwijkingen ook zinvol

geïnterpreteerd kunnen worden. Een belangrijke vraag in dezen is: kunnen VLI's waarin dynamische VRI's zijn opgenomen van een adequaat remplaceschema worden voorzien?

- Zijn de strategieën IHG en OBR in de praktijk bruikbaar? Zijn de strategieën niet te ingewikkeld? Ontstaan er, bijvoorbeeld, fouten omdat de remplaceschema's niet eenvoudig uitvoerbaar zijn? Of is er juist behoefte aan veel preciezere schema's?

- Welke gegevens zou een BOS moeten kunnen verstrekken naast de remplaceschema's? Welke functionaliteit ontbreekt aan het prototype? Welke functionaliteit wordt niet beschreven door de specificatie?

- Welke controles zou men willen uitvoeren op historische gegevens over het falen van lampen? Welke gegevens zou een statistisch registratiesysteem hiertoe moeten verzamelen?

- De programmatuur maakt gebruik van benaderingen van bepaalde functies. Deze benaderingen zijn niet gegarandeerd continue functies. Op discontinuïteiten kunnen eventueel problemen ontstaan als benaderingen geïntegreerd worden of andere operaties ondergaan. Dit punt verdient nadere aandacht bij het uitbouwen van het prototype naar een volwaardig product.

De tweede lijn heeft een meer theoretisch karakter. De volgende vragen en punten zijn interessante en belangrijke onderwerpen voor verder onderzoek.

- In de praktijk neemt het uitvoeren van een remplaceopdracht tijd, alleen al om die reden zal een opdracht nimmer exact op het gevraagde moment uitgevoerd worden. Welke afwijkingen van het gevraagde moment zijn acceptabel? Dat wil zeggen, welke afwijkingen zijn verwaarloosbaar in die zin dat schattingen voor exploitatie en betrouwbaarheid overeind blijven?

- In (8) wordt een fout gemaakt omdat de correctiefactor $\Delta(\underline{k})$ uit (7) is weggelaten. Men zou de heuristiek kunnen proberen te verbeteren door:

  - Voorwaarden te formuleren op de kostenparameters zodat $C_{IHG}(T, \underline{k})$ altijd zijn minimale waarde bereikt voor een paar $\langle T^*, \underline{k}^* \rangle$ met de eigenschap dat $min_{j=1}^{m}(k_j^*) = 1$.
  - Een snelle benadering voor $\Delta(\underline{k})$ te vinden, zodanig dat optimalisatie op basis van (7) kan gebeuren. Een probleem hiermee is dat de kostenfunctie niet separabel meer is. Dit heeft consequenties voor de algoritmiek.
  - Een oplossing met $min_{j=1}^{m}(k_j) = 1$ te forceren. Er is geëxperimenteerd met de volgende operatie. Gegeven T en $\underline{k}$. Als T en $\underline{k}$ niet meer wijzigen en $min_{j=1}^{m}(k_j) \neq 1$, forceer dan een $\underline{k}'$ op de volgende wijze.

$$k'_j = \begin{cases} \frac{k_j}{min_{j=1}^{m}(k_j)} & \text{als } min_{j=1}^{m}(k_j) \text{ een deler is van } k_j \\ \lceil \frac{k_j}{min_{j=1}^{m}(k_j)} \rceil & \text{als } h'_{j,T}(k_j) < 0 \\ \lfloor \frac{k_j}{min_{j=1}^{m}(k_j)} \rfloor & \text{als } h'_{j,T}(k_j) \geq 0 \end{cases}$$

    Hierin is $h'_{j,T}(k_j)$ de afgeleide naar k_j van $h_j(T, k_j)$ bij vaste T. Het forceren van een oplossing op deze wijze is uiteindelijk niet gebruikt omdat de consequenties op dit moment onvoldoende begrepen zijn.

- In de lemma's uit Paragraaf 4.2.1 over IHG wordt gebruik gemaakt van vrij sterke aannamen over de vernieuwingsfuncties M_j en de afgeleiden hiervan. In de praktijk hoeven deze voorwaarden niet te gelden, zelfs niet op het interval $[0, \mu_j]$. Het probleem is dat de huidige voorwaarden niet te controleren zijn. Niet voldoen aan de voorwaarden kan betekenen dat een niet optimale oplossing wordt geretourneerd, het is derhalve relevant om te proberen preciezere voorwaarden voor M_j en m_j te formuleren die controleerbaar zijn.

- De strategie uit Paragraaf 6.4 zou uitgewerkt kunnen worden en vergeleken kunnen worden met OBR en IHG. Zie ook [DS91].
- Het algoritme voor OBR is geformuleerd onder Hypothese 6.5. Zijn er precieze eisen te formuleren die 'run-time' te controleren zijn waaronder Lemma 6.4 geldt. Zijn er precieze eisen te formuleren die 'run-time' te controleren zijn waaronder de eigenschappen zoals geformuleerd in de hypothese gelden? Het gaat hier om eisen die afdwingen dat er één en precies één minimum van C_j is op het interval $[0, \mu_j]$.
- Groepsindeling is bij de huidige stand van zaken een proces waarbij het systeem geen ondersteuning biedt. Een paar vuistregels voor groepsformatie zijn geformuleerd in Paragraaf 8.2. Is hier echter een strategie, eventueel voorzien van een optimalisatiealgoritme, denkbaar? Bijvoorbeeld, 'eerst op lampniveau optimaliseren, en dan aggregeren'.
- In Paragraaf 9 is een aanzet gegeven voor overkoepelende strategieën welke een vervangingssystematieken voor meer dan één VLI specificeren, zogenaamde aggregatiestrategieën. Deze aggregatiestrategieën zouden kunnen worden uitgewerkt door een wiskundig model te ontwikkelen en een optimalisatiealgoritme te ontwikkelen.
- De Aannamen 6.1, 6.2 en 6.3 moeten door uitgebreide simulatie getest worden op hun houdbaarheid in de praktijk. Een aanzet hiertoe wordt gegeven in [Vos95].
- In de voorbeeldjes die wij hebben doorgerekend schijnt altijd het volgende te gelden. Gegeven een groepsindeling, een kostenstructuur en een schema gegeneereerd met IHG. Met OBR is geen schema te genereren, ook niet door de strafkosten aan te passen, dat even betrouwbaar is én goedkoper. Is IHG werkelijk een betere strategie? Is er een tegenvoorbeeld te construeren?
- De basiskosten A zijn een vereenvoudiging van de werkelijkheid. Feitelijk is het aantal wijzen waarop kosten gecombineerd kunnen worden veel groter: aan elke deelverzameling van een verzameling groepen $S_1 \ldots S_m$ kan een post 'gedeelde kosten' A_g, $1 \leq g \leq 2^m$, worden toegekend. In het huidige model is elke A_g simpelweg gelijk aan A. Het is duidelijk dat een dergelijke verfijning de berekeningen veel tijdrovender en misschien wel onuitvoerbaar maakt. Misschien zijn er echter wel bepaalde verfijningen van de modellen of bepaalde richtlijnen op te stellen die het mogelijk maken nauwkeuriger met gedeelde kosten om te kunnen gaan. Een typisch voorbeeld is de situatie waarin voor sommige groepen een hoogwerker nodig is en voor andere niet.
- Hanscom en Cléroux [HC75] tonen numeriek voor blockreplacement aan dat het eerste minimum vanaf de oorsprong het globale minimum is onder zeer milde condities. De kostenfunctie voor IHG lijkt bij vaste $\underline{k}$ hetzelfde gedrag te tonen, zie bijvoorbeeld de vele figuren. Onder welke omstandigheden geldt dat het eerste minimum globaal is?

Referenties

[BW90] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Cambridge Tracts in Theoretical Computer Science 18. Cambridge University Press, 1990.

[Dag82] J.S. Dagnapur. Formulation of a multi item single supplier inventory problem. *Journal of the Operational Research Society*, 33(3):285–286, 1982.

[DS90] R. Dekker and E. Smeitink. A simple approximation to the renewal function. *IEEE Transactions on Reliability*, 39(1):71–75, 1990.

[DS91] R. Dekker and E. Smeitink. Opportunity-based block replacement. *European Journal of Operational Research*, 53:46–63, 1991.

[DSL92] R. Dekker, A. Smit, and J. Losekoot. Combining maintenance activities in an operational phase. *IMA Journal of Mathematics Applied in Business and Industry*, 3:315–332, 1992.

[Ger89] I.B. Gertsbakh. *Statistical Reliability Theory.* Marcel Dekkers, New York, 1989.

[Haa93] Nederland Haarlem. Interview met de heer Thomassen van het GEB Capelle en Krimpen aan den IJssel, 1993.

[HC75] M.A. Hanscom and R. Cléroux. The block replacement problem. *Journal of Statistical Computation and Simulation*, 3:233–248, 1975.

[Kro94] J.J.M.T. Kroone. Functionele specificatie lampremplace. Notitie 940117_1.fsp, versie 2, Maart 1994.

[MM95] S. Mauw and J.C. Mulder. A PSF library of data types. In *ASF+SDF'95*, 1995. Amsterdam.

[MV93] S. Mauw and G.J. Veltink, editors. *Algebraic specification of communication protocols.* Cambridge Tracts in Theoretical Computer Science 36. Cambridge University Press, 1993.

[PFTV90] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in Pascal.* Cambridge University Press, 1990.

[Tij95] H.C. Tijms. *Stochastic Models; an algorithmic approach.* Wiley, New York, 1995.

[Vos95] S.L.E. Vos de Wael. Strategieën voor lampremplace. Master's thesis, Tilburg University, 1995.

[vW93] J.J. van Wamel. A library for PSF. Report P9301, Programming Research Group, University of Amsterdam, 1993.

Deel II
Specificatie

In dit deel van het verslag wordt een specificatie, in PSF, gegeven en besproken die een aanzet probeert te zijn tot een specificatie van een beslissingsondersteunend systeem voor lampremplace. De reden om een specificatie te geven is dat een BOS niet louter optimalisaties voor IHG en OBR uitvoert, maar een heel skala van andere vragen moet kunnen beantwoorden en vele gegevens moet beheren. Om een aanvang te maken met het in kaart brengen van deze materie is deze specificatie gegeven. De specificatie is met enige moeite executeerbaar, kleine toevoegingen en aanpassingen aan de specificatie, speciaal t.b.v. simulatie, helpen het simuleren vereenvoudigen. Voorbeelden van dergelijke toevoegingen en aanpassingen zijn integraal opgenomen in dit Deel, zie Paragraaf 15.4. Voor informatie over PSF wordt de lezer verwezen naar de literatuur [MV93] en electronische archieven (het WWW archief van de vakgroep Formele Methoden van de fac. Wiskunde en Informatica van de Technische Universiteit Eindhoven, of het ftp archief aldaar: `win.tue.nl` in `pub/psf`).

De specificatie wijkt op enkele punten af van het theoretische betoog in Deel I. Ten eerste, de specificatie werkt met discrete tijdstappen in plaats van continue tijd, zie Paragraaf 12. Ten tweede, In het theoretisch model wordt gekeken naar de te ondernemen actie in antwoord op het falen van één lamp. In de praktijk is het heel wel mogelijk dat meerdere defecten van één vli in één keer gemeld worden, zo ook in de specificatie. Hoe men met meedere defecten om moet gaan ligt voor de hand: bepaal de actie per defect en bundel deze acties zodanig dat van twee of meer keer gegenereerde acties slechts een actie overblijft in de bundeling. De afwijking ten opzichte van het theoretisch model komt tot uiting in de kosten. Een dergelijke bundeling van acties betekent dat in de praktijk de kosten lager kunnen uitvallen dan het theoretisch model aangeeft, bijvoorbeeld omdat de basiskosten maar één keer uitgegeven worden in een bundeling.

11 De specificatie in grote lijnen

De specificatie bestaat uit een aantal datamodulen en een aantal procesmodulen. In de datamodulen zijn eenvoudige datatypen (bijv. `LichtpuntLocatie`) en samengestelde datatypen (bijv. `Vli`) gespecificeerd. Tevens zijn operaties die een sleutelrol spelen in de beslissingsondersteuning gespecificeerd, met name, `bepaal-actie` en `preventieve-opdrachten` (optimalisatie speelt als specificatievraagstuk een ondergeschikte rol in dit deel van het verslag). De eerste berekent welke acties moeten worden ondernomen als een lamp in een vli faalt, de tweede berekent een lijst met preventieve vervangingsopdrachten. Deze functies, en vele andere die later besproken worden, worden aangeroepen door een achttal processen die het BOS vormen. Deze processen worden hieronder kort besproken.

Twee ondersteunende processen zijn de `Klok` en het `Interface`. Over de rol van het proces `Klok` wordt verwezen naar Paragraaf 12, het interface is een rudimentaire voorstelling van de interactie met een gebruiker.

De database `Database` beheert twee centrale structuren, een lijst met uit te voeren remplaceopdrachten en een lijst met gebeurtenissen, aan deze lijsten zal hieronder vaak gerefereerd worden met de naam `opn` resp. `gbs`. De lijst met gebeurtenissen is een historisch overzicht van alle relevant geachte activiteiten die door het BOS behandeld zijn, bijvoorbeeld, het in beheer nemen van een vli en het uitgevoerd zijn van een remplaceopdracht. De database kan bepaalde vragen over de structuren beantwoorden en tevens bepaalde operaties uitvoeren, beide op verzoek van de processen die hieronder besproken worden.

Het proces `Configuratie-Analyse` modelleert het 'editen' van de gegevens van een vli, hetzij ten einde een vli toe te voegen, hetzij om de gegevens van een vli te wijzigen. Beide operaties worden uitgevoerd in samenspraak met `SchemaBeheer`. Het optimaliseren van schema's is ook een taak van `Configuratie-Analyse`. Deze taak is echter niet nader uitgewerkt, immers, daar is in Deel I al ruim

aandacht aan besteed; er is slechts een functie, `schema`, die gegeven een vli een schema oplevert, zie module `Schema` in Paragraaf 15.1 en de axioma's ten behoeve van simulatie in Paragraaf 15.4.

Het proces `SchemaBeheer` speelt een rol bij het starten en stoppen van het beheer van een vli. Het kijkt naar de gevraagde data waarop iets zou moeten gebeuren en kijkt naar de consequenties voor de planning, dit in samenspraak met `RemplacePlanning`.

`RemplacePlanning` is het proces dat de planning van remplaceopdrachten beheert. Het voegt bijvoorbeeld autonoom preventieve opdrachten toe aan de planning voor vli's onder IHG.

Het proces `BeslissingsOndersteuning` bepaalt welke actie moet worden ondernomen in antwoord op een melding van een defect.

Als laatste, `Statistische-Analyse` is een zeer rudimentair gemodelleerd proces dat analyse van de gegevens in `gbs` voorstelt.

In Figuur 19 is aangegeven uit welke modulen de specificatie is opgebouwd en hoe de modulen aan elkaar gerelateerd zijn middels de importrelatie. Modulen uit de PSF-library [MM95] (dit is een herziene versie van de library van van Wamel [vW93]) zijn niet opgenomen in de figuur. Merk op dat sommige modulen meerdere malen voorkomen, dit ten einde de graaf planair te maken, deze herhaalde referenties zijn gemerkt met een `*`.

11.1 Executeren van de specificatie

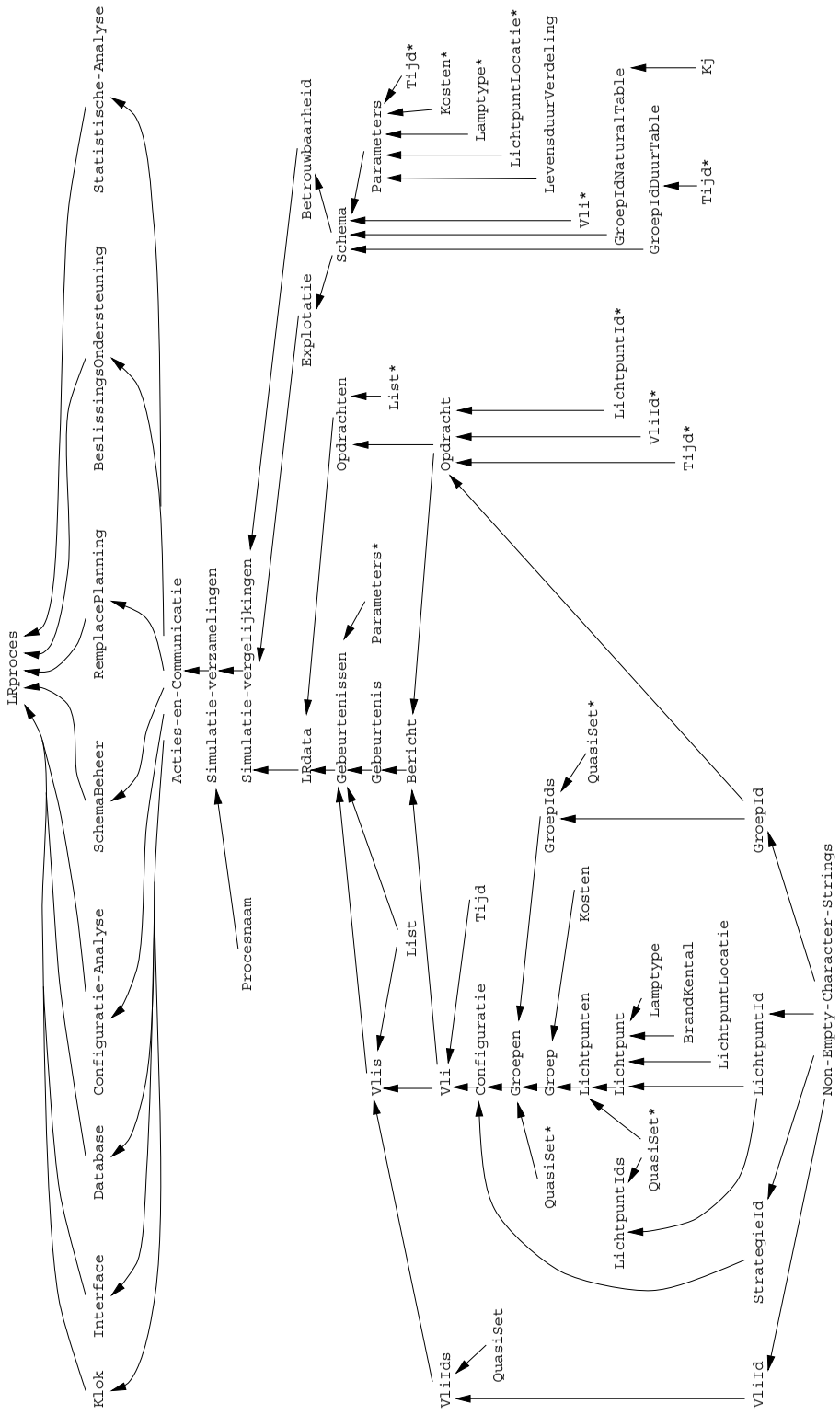
Het is in principe mogelijk om PSF specificaties te executeren. Om practische redenen worden echter vaak een aantal voorzieningen in de specificatie getroffen alvorens tot executie kan worden overgegaan. In dit geval zijn twee voorzieningen getroffen. Ten eerste, soorten die 'veel' objecten bevatten en waarover gesommerd wordt zijn vervangen door overzichtelijke deelverzamelingen. Zo komt men in de procesmodulen extensies '`-f`' tegen bij de soortnamen waarmee naar de deelverzameling verwezen wordt in plaats van de totale verzameling van objecten; merk op dat `-f` geen betekenis heeft voor PSF, het is slechts onze manier om namen te geven aan de deelverzamelingen. De deelverzamelingen zijn gedeclareerd in `Simulatie-verzamelingen` in Paragraaf 15.4. Voegt men dergelijke verzamelingen toe, dan biedt de simulator een keuze aan uit de mogelijke objecten, in het andere geval zal de gebruiker een term moeten intikken; als dit tijdens een simulatie-run herhaald moet gebeuren is dat heel vervelend, vandaar. Ten tweede, enkele functies zijn niet uitgespecifieerd, dit zijn `schema`, `exploitatie` en `betrouwbaarheid`. Om toch te kunnen simuleren zijn voor een aantal vli's axioma's opgenomen die voor precies die vli's een waarde voor deze functies definiëren, zie module `Simulatie-vergelijkingen` in Paragraaf 15.4.

12 Tijd in de specificatie

In deze paragraaf worden een aantal onderdelen van de specificatie in meer datail besproken die gerelateerd zijn aan het voortschrijden van de tijd.

12.1 Tijd

Tijd, zoals we dat gewend zijn, komt ons voor als een continu voortschrijdend proces. Alle acties nemen tijd, en tussen twee punten in de tijd is dan ook maar een beperkte hoeveelheid activiteit in sequentie mogelijk. In PSF zijn deze eigenschappen niet adequaat te modelleren, de enige mogelijkheid lijkt het toevoegen van een proces, dat we bijvoorbeeld `Klok` noemen, dat discrete tijdstappen zet en deze telt. Tussen twee tijdstappen is het, zonder bepaalde ingrepen, mogelijk een willekeurig aantal acties in sequentie uit te voeren. Wil men er voor zorgen dat bepaalde processen bepaalde acties uitvoeren 'op tijd' dan moeten de klok en deze processen weet van elkaar hebben en synchroniseren. Dergelijke processen kunnen dan eventueel het voorschrijden van de 'tijd' — immers slechts een PSF proces — beïnvloeden. Een aantal tijd gerelateerde zaken uit de specificatie worden hieronder besproken.



Figuur 19: Importrelatie van de data- procesmodulen.

12.2 De soorten Datum, Duur0 en Duur1

Zoals gezegd werkt de specificatie met discrete tijd. Deze tijd begint te lopen op het moment `startdatum`, een constante van het type `Datum`, die is gespecificeerd in de module `Parameters`, zie Paragraaf 15.4. Een datum is dus een naam van een tijdstip, eigenlijk een naam van een interval, op de 'tijdas'. Een duur is de afstand tussen tijdstippen. We onderscheiden twee duren. Duren die groter gelijk 0 zijn, deze zitten in de soort `Duur0`, en duren die groter gelijk 1 zijn, deze zitten in soort `Duur1`. De reden om twee duren in te voeren is dat duren die gebruikt worden in vervangingsschema's aan de eigenschap voldoen dat ze groter of gelijk 1 zijn, een basiscyclus, T, van 0 tijdseenheden voor een IHG schema is immers zinloos.

12.3 Het uitrekenen van preventieve opdrachten

De functie `preventieve−opdrachten(dat, dat′, gbs)` bepaalt preventieve remplaceopdrachten die in de tijdspanne $[$`dat, dat′`$\rangle$ moeten worden uitgevoerd. Gegevens over de vli's die in beheer zijn in dit interval, of in een gedeelte van dit interval, worden gehaald uit `gbs`. De functie `preventieve-opdrachten` is van belang voor het proces `RemplacePlanning` en wordt op diens verzoek geëxecuteerd door het proces `Database`.

12.4 Het bepalen van te ondernemen actie ingeval van een defect

De functie `bepaal-actie(vid,lids,dat,gbs)` bepaalt aan de hand van een defect — de lampen in de lijst `lids` van vli met naam `vid` zijn gefaald op tijstip `dat` — welke onderhoudsactiviteit moet worden ondernomen. Gegevens over de vli worden uit `gbs`, de lijst met gebeurtenissen, gevist. In geval van IHG levert de functie een lijst met correctieve opdrachten, één voor elke lamp die defect is gemeld. In geval van OBR moet van elke groep van `vid` opgezocht worden wanneer de groep voor het laatst preventief is vervangen. Immers, als dit tijdstip + de 't_j' voor de groep kleiner of gelijk het huidige tijdstip is, dan moet de groep preventief worden vervangen, zie Paragraaf 6. Voor defecten die niet opgaan in een preventieve vervanging wordt een correctieve vervangingsopdracht gegenereerd. Indien geen bericht van een uitgevoerde preventieve vervanging voor een zekere groep wordt gevonden in de gebeurtenissen administratie, dan wordt `startdatum` gekozen. Dit is een acceptabele keuze, als een preventieve opdracht toevallig nog in uitvoering is wordt er hoogstens één teveel gegenereerd, dat is beter dan één te weinig. De functie `bepaal-actie` is van belang voor het proces `BeslissingsOndersteuning` en wordt op verzoek van dit proces uitgevoerd door het proces `Database`.

12.5 De functie onder-schema

De functie `onder-schema(vid,dat,gbs)` geeft antwoord op de vraag of een vli met naam `vid` op een zeker tijdstip `dat` in beheer is. Dit is het geval als:

- Er in de gebeurtenissen een bericht `start(vli, dat′)`, met `vid(vli) = vid`, en er geen bericht `stop(vid, dat″)` is dat na het start bericht ligt en `dat′ .lte. dat` = `true`.
- Er in de gebeurtenissen een bericht `start(vli, dat′)`, met `vid(vli) = vid`, en een bericht `stop(vid, dat″)` zijn geregisteerd zonder tussen liggende start en stop berichten en tevens dat `and(dat′ .lte. dat, dat .lt. dat″)` = `true`.

De functie `onder-schema` wordt uitgevoerd door het proces `Database` op verzoek van het proces `BeslissingsOndersteuning`.

12.6 RemplacePlanning

Er is expliciete synchronisatie tussen `Klok` en `RemplacePlanning`. Men wil immers dat de planning altijd up-to-date is: de klok mag niet vooruit lopen zonder dat de planning wordt bijgewerkt met uit te voeren preventieve onderhoudsacties. Er geldt dat er altijd een verfrissing van de planning plaatsvindt vóór het tijdstip tot waar de planning bijgewerkt is. In de specificatie wordt dit gecontroleerd met de test: `and(nu .lt. tot, tot .lt. (nu + horizon + een))`. Evalueert deze test naar `true` dan moet de planning bijgewerkt worden. Door de `Klok` en `RemplacePlanning` expliciet te laten synchroniseren op het voortschrijden van de tijd wordt afgedwongen dat een update iedere tijdstap plaatsvindt. Als de test faalt, dat wil zeggen de expressie evalueert naar `false`, zijn er twee gevallen van belang: `nu .lt. tot` evalueert naar false, in dat geval faalt het proces `RemplacePlanning`, de planning is immers onherstelbaar achter; `tot .ge. (nu + horizon + een)` is waar, in dat geval is de planning up-to-date en hoeft er niets te gebeuren.

Het proces `BOS` in de module `LRproces` kiest een `Horizon` en start `RemplacePlanning` met als argumenten `dat + Horizon + een` en `Horizon`. Merk op dat `Horizon` een element is uit `Duur1` en derhalve altijd groter gelijk 1 tijdstap is. Omdat bij het starten van het BOS nog geen vli's in beheer zijn is de planning dus bijgewerkt tot `dat + Horizon + een`.

12.7 SchemaBeheer

Als het beheer van een vli gestaakt wordt op een zeker tijdstip, dan is het redelijk te eisen uit overwegingen van consistentie dat dit tijdstip ligt ná de als laatste uitgevoerde remplace voor deze vli. Anders was de vli kennelijk nog wel in beheer. In de module `SchemaBeheer` wordt hiertoe gebruik gemaakt van `tijd-laatst-uitgevoerde-opdracht : Vlid # Gebeurtenissen -> Datum`. Indien er in `gbs` geen opdracht te vinden is die ten behoeve van de behandelde vli is uitgevoerd levert deze functie `startdatum` op. Dit is acceptabel omdat een vli niet gestopt wordt voordat hij gestart is. Een vli start op zijn vroegst op `startdatum`, derhalve ligt de stop dan altijd later.

13 Invarianten

Bij het functioneren en bij de opbouw en het onderhoud van de structuren `gbs` en `opn` door de verschillende processen zou het zo moeten zijn dat een aantal eigenschappen worden gewaarborgd. Deze eigenschappen worden in deze paragraaf uitgewerkt. De eigenschappen zouden invarianten van het BOS moeten zijn, dit is echter niet geverifieerd.

- Het proces BOS bevat geen deadlock of lifelock.
- Als een vli gestopt is, dan is deze ook gestart op een strikt eerder moment. Om precies te zijn, als `stop(vid,t)` in `gbs` voorkomt, dan is er eerder in `gbs` een registratie $\texttt{start}(\texttt{vli}(\texttt{vid}, \texttt{cf}, \texttt{t}'))$ opgenomen en $\texttt{t}'\ \texttt{.lt.}\ \texttt{t} = \texttt{true}$. Tevens komen de gebeurtenissen `start` en `stop` geordend en alternerend voor van begin naar eind van de rij registraties: eerst `start`, daarna `stop` et cetera.
- Defecten komen eenmalig voor in `gbs`, dat wil zeggen, als `defect(vi,lids,t)` voorkomt in de lijst met gebeurtenissen `gbs` dan is er geen defect $\texttt{defect}(\texttt{vi}, \texttt{lids}', \texttt{t})$ in `gbs` zodanig dat `lids` en $\texttt{lids}'$ een lamp gemeenschappelijk hebben.
- Als er een defect `defect(vi,lids,t)` geregistreerd is in `gbs` dan is `vi` in beheer op `t`, dus er is een gebeurtenis $\texttt{start}(\texttt{vli}(\texttt{vid}, \texttt{cf}, \texttt{t}'))$ en $\texttt{t}'\ \texttt{.lte.}\ \texttt{t} = \texttt{true}$; als `vid` niet meer in beheer is op $\texttt{t}''$ wat blijkt uit de registratie in `gbs` van een gebeurtenis $\texttt{stop}(\texttt{vid}, \texttt{t}'')$, dan geldt tevens $\texttt{t}\ \texttt{.lt.}\ \texttt{t}'' = \texttt{true}$.
- Een vli in `gbs` voldoet aan de volgende eigenschappen:

- één of meer groepen;
- elke groep bevat één of meer lampen;
- elke groep heeft een unieke naam binnen deze vli;
- elke lamp heeft een unieke naam binnen deze vli;

- Alle opdrachten in de lijst `opn` gaan over vli's die op het moment dat de opdracht zou moeten worden uitgevoerd onder schema waren.
- Opdrachten zijn overeenkomstig het schema van de vli en de theorie. Bijvoorbeeld, voor vli's onder IHG is er een groep met '$k_j = 1$'.
- Bij in beheername van een vli worden preventieve opdrachten voor alle groepen gegenereerd.

14 Tekortkomingen van de specificatie

De discussie over de functionaliteit van het BOS is niet afgerond, de specificatie is dan ook niet meer dan een aanzet tot een ontwerp. In deze paragraaf worden punten opgesomd die aanpassing, uitbreiding of nadere studie behoeven.

- Er lijkt behoefte te bestaan aan een functie die besluit of de afwijking tussen een daadwerkelijke vervangmoment en een een gepland vervangmoment acceptabel is. Deze functie wordt in de specificatie `inmarge` genoemd, zie module Gebeurtenissen in Paragraaf 15.2, de huidige definitie van `inmarge` luidt:

  $$\texttt{inmarge}(\texttt{t}, \texttt{op}, \texttt{gbs}) \;=\; \texttt{eq}(\texttt{t}, \texttt{tijd}(\texttt{op})).$$

  Deze definitie is een sterke vereenvoudiging van de werkelijkheid. Een vervanging wordt nimmer exact op het door het BOS gevraagde moment geheel uitgevoerd; alleen al omdat een vervanging tijd kost. Welke afwijking van het gevraagde tijdstip besliskundig acceptabel is zal nader moeten worden vastgesteld. Tevens zal moeten worden vastgesteld welke actie ondernomen moet worden indien een afwijking geconstateerd wordt.

  Als men `inmarge` ruimer definieert, dat wil zeggen een afwijking toestaat van het geplande vervangmoment waarop een vervanging ook mag plaatsvinden, heeft dit consequenties voor het genereren van opdrachten zoals gedaan door `preventieve-opdrachten(.)`. Opdrachten die namelijk eerder dan gepland worden uitgevoerd moeten niet bij een verversing van de lijst van opdrachten wederom toegevoegd worden, dat wordt nu niet getest.

- De `Database` beheert twee structuren, een lijst van gebeurtenissen, hier `gbs` genoemd, en een lijst van opdrachten, hier `opn` genoemd. Er is wat voor te zeggen om een historisch overzicht van alle gebeurtenissen te hebben, dit met het oog op bijvoorbeeld analyse van het functioneren van het BOS of het reconstrueren van de uitvoering der opdrachten. Echter, waar het betreft het genereren van acties in antwoord op defecten en het genereren van preventieve opdrachten is `gbs` geen handige structuur, het is te veel een vergaarbak. Tevens zijn er een aantal consistentie-eisen te formuleren met betrekking tot specifieke gebeurtenissen en de lijst met opdrachten; elegante definitie van deze eisen nopen tot een andere structuur van de database. Waarschijnlijk is het beter een lijst te maken met entries per vli, per entry worden dan relevante gebeurtenissen uit het leven van een vli en de lopende opdrachten geadministreerd. Over de opdrachten zijn nog enkele andere opmerkingen de maken die hieronder volgen.

- `Opdrachten` is een type van lijsten van correctieve en preventieve opdrachten. De interpretatie van dergelijke lijsten is niet altijd eenduidig omdat er een afhankelijkheid bestaat van de gevoerde strategie. Het is evident dat twee dezelfde opdrachten als één opdracht dienen te worden

geïnterpreteerd. Voor OBR zou men echter ook wensen dat geldt:

$$\texttt{preventief}(\texttt{vid}, \texttt{gid}, \texttt{dat}) \sim (\texttt{preventief}(\texttt{vid}, \texttt{gid}, \texttt{dat}') \sim \texttt{opn}) = \texttt{preventief}(\texttt{vid}, \texttt{gid}, \texttt{min}(\texttt{dat}, \texttt{dat}')) \sim \texttt{opn}$$

Een axioma dat echter voor IHG onjuist is. Dit pleit ervoor per strategie lijsten met opdrachten te definiëren als aparte soorten, of op een andere wijze informatie over de strategie ter beschikking te maken.

- Een lastig probleem met opdrachten is de combinatie van opdrachten. Gegeven twee lijsten `opn` en `opn`$'$. Wat is de concatenatie `opn` $\sim$ `opn`$'$? Stel er wordt gemeld dat lamp ℓ defect is. Dit resulteert in een of meer opdrachten die op de lijst `opn` van het proces `Database` geplaatst worden. Deze opdrachten worden in uitvoering genomen. Nu wordt weer gemeld dat ℓ defect is. Is dit hetzelfde defect? Zo ja, dan kan het genegeerd worden. Zo nee, hoe ver is de uitvoering van het vorige defect gevorderd? Welke opdrachten zijn uitgevoerd, en welke niet? Kunnen er combinaties getroffen worden. Kortom, de operationele aspecten die een rol spelen bij de beslissingsonderstening en het onderhouden van een lijst met opdrachten is complex. In de specificatie, en met name in het proces `BeslissingsOndersteuning` is van de bovengeschetste probematiek geabstraheerd.

- Het proces `RemplacePlanning` zou moeten waarschuwen als een opdracht niet wordt uitgevoerd. Of, althans, als de uitvoering niet gemeld wordt binnen een tijdsbestek dat door `inmarge` als acceptabel wordt aangemerkt.

- Zij c een correctieve opdracht en p een preventieve opdracht die ook de lamp uit c bevat. De gewenste uitvoerdatum van p ligt na die van c. De tijd schrijdt voort en p wordt uitgevoerd. Dan zou c automatisch kunnen vervallen. Dat gebeurt nu niet.

- Het proces `Statistische-Analyse` is volstrekt rudimentair.

- De edit-, selectie-, en rapportagefaciliteiten die het BOS zou moeten bieden zijn niet of in onvoldoende mate gespecificeerd.

- Opdrachten voor een zekere vli die op dezelfde datum dienen te worden uitgevoerd zouden gebundeld kunnen worden tot één opdracht.

- Er zijn geen prestatieeisen geformuleerd.

15 Specificatie

In deze Paragraaf zijn alle modulen van de simuleerbare specificatie afgedrukt met weglating van de modulen uit de PSF-library [MM95]. Superscripten bij modulennamen in de importsecties zijn paginanummer verwijzingen.

15.1 Elementaire datatypen

Deze paragraaf bevat specificaties van datastructuren wier bestaansrecht en eigenschappen in zekere zin boven discussie verheven zijn. In de daarop volgende paragrafen komen dan structuren aan de orde waar dat minder duidelijk voor geldt, vandaar een scheiding.

Allereerst komen aan bod zeer eenvoudige soorten als lichtpuntlocaties, `LichtpuntLocatie`; namen (*Id*entifiers) van allerlei zaken, bijv `LichtpuntId`, en `StrategieId`; generieke structuren als lijsten, `List`. Vervolgens worden meer complexe structuren opgebouwd, bijv., de groep `Groep`, lijsten van groepen `Groepen`, het schema `Schema`, de vli `Vli`, lijsten van vli's `Vlis`.

data module *List*
begin
  parameters
    Item
    begin
      sorts *L-ELEMENT*
      functions
        $eq : L\text{-}ELEMENT \# L\text{-}ELEMENT \rightarrow BOOLEAN$
    end *Item*
  exports
  begin
    sorts *List*
    functions
      $leeg : \rightarrow List$
      $\_ \sim \_ : L\text{-}ELEMENT \# List \rightarrow List$
      $\_ \sim \_ : List \# List \rightarrow List$
      $element : L\text{-}ELEMENT \# List \rightarrow BOOLEAN$
      $verwijder : L\text{-}ELEMENT \# List \rightarrow List$
      $eq : List \# List \rightarrow BOOLEAN$
  end
  imports *Booleans*
  variables
    $e1, e2 : \rightarrow L\text{-}ELEMENT$
    $l1, l2 : \rightarrow List$
  equations
  [l1] $leeg \sim l1 = l1$
  [l2] $(e1 \sim l1) \sim l2 = e1 \sim (l1 \sim l2)$
  [e1] $element(e1, leeg) = false$
  [e2] $element(e1, e2 \sim l2) = or(eq(e1, e2), element(e1, l2))$
  [r1] $verwijder(e1, leeg) = leeg$
  [r2] $verwijder(e1, e2 \sim l1) = l1$ **when** $eq(e1, e2) = true$
  [r3] $verwijder(e1, e2 \sim l1) = e2 \sim verwijder(e1, l1)$ **when** $eq(e1, e2) = false$
  [e1] $eq(leeg, leeg) = true$
  [e2] $eq(e1 \sim l1, leeg) = false$
  [e3] $eq(leeg, e1 \sim l1) = false$
  [e4] $eq(e1 \sim l1, e2 \sim l2) = and(eq(e1, e2), eq(l1, l2))$
end *List*

data module *LichtpuntLocatie*
begin
  exports
  begin
    sorts *LichtpuntLocatie*
    functions
      $laag : \rightarrow LichtpuntLocatie$
      $hoog : \rightarrow LichtpuntLocatie$
      $eq : LichtpuntLocatie \# LichtpuntLocatie \rightarrow BOOLEAN$
  end
  imports *Booleans*
  equations

[eq1] $eq(laag, laag) = true$
[eq2] $eq(laag, hoog) = false$
[eq3] $eq(hoog, laag) = false$
[eq4] $eq(hoog, hoog) = true$

end *LichtpuntLocatie*

data module *Tijd*
begin
exports
begin
sorts *Datum*, *Duur0*, *Duur1*
functions

$datum$	$: NATURAL$	$\rightarrow Datum$
$\_ > \_$	$: Datum \# Datum$	$\rightarrow BOOLEAN$
$\_ \geq \_$	$: Datum \# Datum$	$\rightarrow BOOLEAN$
$\_ < \_$	$: Datum \# Datum$	$\rightarrow BOOLEAN$
$\_ \leq \_$	$: Datum \# Datum$	$\rightarrow BOOLEAN$
eq	$: Datum \# Datum$	$\rightarrow BOOLEAN$
$duur0$	$: NATURAL$	$\rightarrow Duur0$
$\_ > \_$	$: Duur0 \# Duur0$	$\rightarrow BOOLEAN$
$\_ \geq \_$	$: Duur0 \# Duur0$	$\rightarrow BOOLEAN$
$\_ < \_$	$: Duur0 \# Duur0$	$\rightarrow BOOLEAN$
$\_ \leq \_$	$: Duur0 \# Duur0$	$\rightarrow BOOLEAN$
eq	$: Duur0 \# Duur0$	$\rightarrow BOOLEAN$
$duur1$	$: NATURAL$	$\rightarrow Duur1$
$\_ > \_$	$: Duur1 \# Duur1$	$\rightarrow BOOLEAN$
$\_ \geq \_$	$: Duur1 \# Duur1$	$\rightarrow BOOLEAN$
$\_ < \_$	$: Duur1 \# Duur1$	$\rightarrow BOOLEAN$
$\_ \leq \_$	$: Duur1 \# Duur1$	$\rightarrow BOOLEAN$
eq	$: Duur1 \# Duur1$	$\rightarrow BOOLEAN$
$\_ + \_$	$: Datum \# Duur0$	$\rightarrow Datum$
$\_ + \_$	$: Datum \# Duur1$	$\rightarrow Datum$
$\_ * \_$	$: NATURAL \# Duur0$	$\rightarrow Duur0$
$\_ * \_$	$: NATURAL \# Duur1$	$\rightarrow Duur0$
max	$: Datum \# Datum$	$\rightarrow Datum$
min	$: Datum \# Datum$	$\rightarrow Datum$
nul	$:$	$\rightarrow Duur0$
een	$:$	$\rightarrow Duur1$

end
imports *Naturals*
variables $n1, n2 : \rightarrow NATURAL$
equations
[d1] $datum(n1) > datum(n2) = gt(n1, n2)$
[d2] $datum(n1) \geq datum(n2) = ge(n1, n2)$
[d3] $datum(n1) < datum(n2) = lt(n1, n2)$
[d4] $datum(n1) \leq datum(n2) = le(n1, n2)$
[d5] $eq(datum(n1), datum(n2)) = eq(n1, n2)$
[d6] $duur0(n1) > duur0(n2) = gt(n1, n2)$
[d7] $duur0(n1) \geq duur0(n2) = ge(n1, n2)$
[d8] $duur0(n1) < duur0(n2) = lt(n1, n2)$

[d9] $duur0(n1) \leq duur0(n2) = le(n1, n2)$
[d10] $eq(duur0(n1), duur0(n2)) = eq(n1, n2)$
[d11] $duur1(n1) > duur1(n2) = gt(n1, n2)$
[d12] $duur1(n1) \geq duur1(n2) = ge(n1, n2)$
[d13] $duur1(n1) < duur1(n2) = lt(n1, n2)$
[d14] $duur1(n1) \leq duur1(n2) = le(n1, n2)$
[d15] $eq(duur1(n1), duur1(n2)) = eq(n1, n2)$
[o1] $datum(n1) + duur0(n2) = datum(n1 + n2)$
[o1] $datum(n1) + duur1(n2) = datum(n1 + n2 + nat(\hat{}1))$
[v1] $n1 * duur0(n2) = duur0(n1 * n2)$
[v1] $n1 * duur1(n2) = duur0(n1 * (n2 + nat(\hat{}1)))$
[ma1] $max(datum(n1), datum(n2)) = datum(n1)$ **when** $ge(n1, n2) = true$
[ma2] $max(datum(n1), datum(n2)) = datum(n2)$ **when** $ge(n2, n1) = true$
[mi1] $min(datum(n1), datum(n2)) = datum(n1)$ **when** $lt(n1, n2) = true$
[mi2] $min(datum(n1), datum(n2)) = datum(n2)$ **when** $le(n2, n1) = true$
[dz1] $nul = duur0(nat(\hat{}0))$
[dz2] $een = duur1(nat(\hat{}0))$

end *Tijd*

data module *Lamptype*
begin
exports
begin
sorts *Lamptype*
functions
$230V75W$: $\rightarrow Lamptype$
$40V40W$: $\rightarrow Lamptype$
$Halogeen10V$: $\rightarrow Lamptype$
eq : $Lamptype \# Lamptype \rightarrow BOOLEAN$
end
imports *Naturals*
functions
$i : Lamptype \rightarrow NATURAL$
variables $lt, lt' : \rightarrow Lamptype$
equations
[i1] $i(230V75W) = nat(\hat{}1)$
[i2] $i(40V40W) = nat(\hat{}2)$
[i3] $i(Halogeen10V) = nat(\hat{}3)$
[eq1] $eq(lt, lt') = eq(i(lt), i(lt'))$
end *Lamptype*

data module *LevensduurVerdeling*
begin
exports
begin
sorts *VormWeibull*, *SchaalWeibull*, *LR-VormErlang*, *SchaalErlang*, *LevensduurVerdeling*
functions
vw : $DIGIT\text{-}SEQ \# DIGIT\text{-}SEQ \rightarrow VormWeibull$
sw : $DIGIT\text{-}SEQ \# DIGIT\text{-}SEQ \rightarrow SchaalWeibull$
ve : $NATURAL \rightarrow LR\text{-}VormErlang$

se : *DIGIT-SEQ* # *DIGIT-SEQ* → *SchaalErlang*
weibull : *VormWeibull* # *SchaalWeibull* → *LevensduurVerdeling*
erlang : *LR-VormErlang* # *SchaalErlang* → *LevensduurVerdeling*

end
imports *Naturals*
variables
d1, *d2*, *d3*, *d4* : → *DIGIT-SEQ*
n : → *NATURAL*
equations
[v1] *weibull*(*vw*(*d1*, *d2*), *sw*(*d3*, *d4*)) = *weibull*(*vw*(*cut-left-zeroes*(*d1*), *d2*), *sw*(*d3*, *d4*))
when *first*(*d1*) = 0
[v2] *weibull*(*vw*(*d1*, *d2*), *sw*(*d3*, *d4*)) = *weibull*(*vw*(*d1*, *d2*), *sw*(*cut-left-zeroes*(*d3*), *d4*))
when *first*(*d3*) = 0
[v3] *erlang*(*ve*(*nat*(^0)), *se*(*d1*, *d2*)) = *erlang*(*ve*(*nat*(^2)), *se*(*d1*, *d2*))
[v4] *erlang*(*ve*(*nat*(^1)), *se*(*d1*, *d2*)) = *erlang*(*ve*(*nat*(^2)), *se*(*d1*, *d2*))
[v5] *erlang*(*ve*(*n*), *se*(*d1*, *d2*)) = *erlang*(*ve*(*n*), *se*(*cut-left-zeroes*(*d1*), *d2*)) **when** *first*(*d1*) = 0
end *LevensduurVerdeling*

data module *Kj*
begin
exports
begin
sorts *Kj*
functions
kj-een : → *Kj*
kj : *NATURAL* → *Kj*
_ * _ : *Kj* # *Duur0* → *Duur0*
_ * _ : *Kj* # *Duur1* → *Duur0*
eq : *Kj* # *Kj* → *BOOLEAN*
end
imports *Tijd*[61]
variables *n1*, *n2* : → *NATURAL*
equations
[m1] *kj*(*n1*) ∗ *duur0*(*n2*) = (*n1* + *nat*(^1)) ∗ *duur0*(*n2*)
[m1] *kj*(*n1*) ∗ *duur1*(*n2*) = (*n1* + *nat*(^1)) ∗ *duur1*(*n2*)
[e1] *eq*(*kj*(*n1*), *kj*(*n2*)) = *eq*(*n1*, *n2*)
end *Kj*

data module *QuasiSet*
begin
parameters
Item
begin
sorts *Q-ELEMENT*
functions
eq : *Q-ELEMENT* # *Q-ELEMENT* → *BOOLEAN*
end *Item*
exports
begin
sorts *QuasiSet*
functions

leeg : $\rightarrow$ *QuasiSet*
_ ~ _ : *Q-ELEMENT* # *QuasiSet* $\rightarrow$ *QuasiSet*
element : *Q-ELEMENT* # *QuasiSet* $\rightarrow$ *BOOLEAN*
verwijder : *Q-ELEMENT* # *QuasiSet* $\rightarrow$ *QuasiSet*
_ ~ _ : *QuasiSet* # *QuasiSet* $\rightarrow$ *QuasiSet*
isect : *QuasiSet* # *QuasiSet* $\rightarrow$ *QuasiSet*
diff : *QuasiSet* # *QuasiSet* $\rightarrow$ *QuasiSet*
subset : *QuasiSet* # *QuasiSet* $\rightarrow$ *BOOLEAN*
eq : *QuasiSet* # *QuasiSet* $\rightarrow$ *BOOLEAN*

end

imports *Naturals*

variables

i1, *i2* : $\rightarrow$ *Q-ELEMENT*
s1, *s2* : $\rightarrow$ *QuasiSet*

equations

[ii1] *element*(*i1*, *leeg*) = *false*
[ii2] *element*(*i1*, *i2* ~ *s1*) = *or*(*eq*(*i1*, *i2*), *element*(*i1*, *s1*))
[R1] *verwijder*(*i1*, *leeg*) = *leeg*
[R2] *verwijder*(*i1*, *i1* ~ *s1*) = *verwijder*(*i1*, *s1*)
[R3] *verwijder*(*i1*, *i2* ~ *s1*) = *i2* ~ *verwijder*(*i1*, *s1*) **when** *eq*(*i1*, *i2*) = *false*
[U1] *leeg* ~ *s1* = *s1*
[U2] (*i1* ~ *s1*) ~ *s2* = *i1* ~ (*s1* ~ *s2*)
[i1] *isect*(*leeg*, *s1*) = *leeg*
[i2] *isect*(*i1* ~ *s1*, *s2*) = *i1* ~ *isect*(*s1*, *s2*) **when** *element*(*i1*, *s2*) = *true*
[i3] *isect*(*i1* ~ *s1*, *s2*) = *isect*(*s1*, *s2*) **when** *element*(*i1*, *s2*) = *false*
[D1] *diff*(*s1*, *leeg*) = *s1*
[D2] *diff*(*s1*, *i1* ~ *s2*) = *verwijder*(*i1*, *diff*(*s1*, *s2*))
[S1] *subset*(*leeg*, *s1*) = *true*
[S2] *subset*(*i1* ~ *s1*, *s2*) = *and*(*element*(*i1*, *s2*), *subset*(*s1*, *s2*))
[eq1] *eq*(*s1*, *s2*) = *and*(*subset*(*s1*, *s2*), *subset*(*s2*, *s1*))

end *QuasiSet*

data module *Non-Empty-Character-Strings*

begin

exports

begin

functions

_ > _ : *CHAR-STRING* # *CHAR-STRING* $\rightarrow$ *BOOLEAN*

end

imports

Non-Empty-Sequences
{*Elements* **bound by** [*ITEM* $\rightarrow$ *CHARACTER*, *eq* $\rightarrow$ *eq*] **to** *Characters*
renamed by [*SEQ* $\rightarrow$ *CHAR-STRING*]}

functions

_ > _ : *CHAR-STRING* # *CHAR-STRING* $\rightarrow$ *BOOLEAN*

variables

cs, *cs'* : $\rightarrow$ *CHAR-STRING*
c1, *c2* : $\rightarrow$ *CHARACTER*

equations

[g0] *cs* > *cs'* = *reverse*(*cs*) > *reverse*(*cs'*)

[g1] $\hat{}c1 > \hat{}c2 = gt(ord(c1), ord(c2))$
[g2] $(cs \hat{} c1) > (cs' \hat{} c2) = or(gt(ord(c1), ord(c2)), and(eq(c1, c2), cs > cs'))$
[g3] $\hat{}c1 > (cs \hat{} c2) = gt(ord(c1), ord(c2))$
[g4] $(cs \hat{} c1) > \hat{}c2 = gt(ord(c1), ord(c2))$
end *Non-Empty-Character-Strings*

data module *VliId*
begin
exports
begin
sorts *VliId*
functions
$vid : CHAR\text{-}STRING \to VliId$
$eq : VliId \# VliId \to BOOLEAN$
$\_ > \_ : VliId \# VliId \to BOOLEAN$
end
imports *Non-Empty-Character-Strings*[64]
variables $cs, cs' : \to CHAR\text{-}STRING$
equations
[e1] $eq(vid(cs), vid(cs')) = eq(cs, cs')$
[g1] $vid(cs) > vid(cs') = cs > cs'$
end *VliId*

data module *VliIds*
begin
imports
QuasiSet[63]
{*Item* **bound by** [*Q-ELEMENT* $\to$ *VliId*, *eq* $\to$ *eq*] **to** *VliId*
renamed by [*QuasiSet* $\to$ *VliIds*, *leeg* $\to$ *0vids*]}
end *VliIds*

data module *LichtpuntId*
begin
exports
begin
sorts *LichtpuntId*
functions
$lid : CHAR\text{-}STRING \to LichtpuntId$
$eq : LichtpuntId \# LichtpuntId \to BOOLEAN$
$\_ > \_ : LichtpuntId \# LichtpuntId \to BOOLEAN$
end
imports *Non-Empty-Character-Strings*[64]
variables $cs, cs' : \to CHAR\text{-}STRING$
equations
[e1] $eq(lid(cs), lid(cs')) = eq(cs, cs')$
[g1] $lid(cs) > lid(cs') = cs > cs'$
end *LichtpuntId*

data module *GroepId*
begin
exports

begin
sorts *GroepId*
functions
gid : *CHAR-STRING* → *GroepId*
eq : *GroepId # GroepId* → *BOOLEAN*
_ > _ : *GroepId # GroepId* → *BOOLEAN*
end
imports *Non-Empty-Character-Strings*[64]
variables *cs*, *cs'* : → *CHAR-STRING*
equations
[eq1] *eq*(*gid*(*cs*), *gid*(*cs'*)) = *eq*(*cs*, *cs'*)
[g1] *gid*(*cs*) > *gid*(*cs'*) = *cs* > *cs'*
end *GroepId*

data module *Opdracht*
begin
exports
begin
sorts *Opdracht*
functions
correctief : *VliId # LichtpuntId # Datum* → *Opdracht*
preventief : *VliId # GroepId # Datum* → *Opdracht*
correctief : *Opdracht* → *BOOLEAN*
tijd : *Opdracht* → *Datum*
vid : *Opdracht* → *VliId*
eq : *Opdracht # Opdracht* → *BOOLEAN*
end
imports *VliId*[65], *Tijd*[61], *LichtpuntId*[65], *GroepId*[65]
variables
dat, *dat'* : → *Datum*
vid, *vid'* : → *VliId*
lid, *lid'* : → *LichtpuntId*
gid, *gid'* : → *GroepId*
equations
[c1] *correctief*(*correctief*(*vid*, *lid*, *dat*)) = *true*
[c2] *correctief*(*preventief*(*vid*, *gid*, *dat*)) = *false*
[t1] *tijd*(*correctief*(*vid*, *lid*, *dat*)) = *dat*
[t2] *tijd*(*preventief*(*vid*, *gid*, *dat*)) = *dat*
[v1] *vid*(*correctief*(*vid*, *lid*, *dat*)) = *vid*
[v2] *vid*(*preventief*(*vid*, *gid*, *dat*)) = *vid*
[e1] *eq*(*correctief*(*vid*, *lid*, *dat*), *correctief*(*vid'*, *lid'*, *dat'*)) =
and(*eq*(*vid*, *vid'*), *and*(*eq*(*lid*, *lid'*), *eq*(*dat*, *dat'*)))
[e2] *eq*(*preventief*(*vid*, *gid*, *dat*), *preventief*(*vid'*, *gid'*, *dat'*)) =
and(*eq*(*vid*, *vid'*), *and*(*eq*(*gid*, *gid'*), *eq*(*dat*, *dat'*)))
[e3] *eq*(*correctief*(*vid*, *lid*, *dat*), *preventief*(*vid'*, *gid'*, *dat'*)) = *false*
[e4] *eq*(*preventief*(*vid'*, *gid'*, *dat'*), *correctief*(*vid*, *lid*, *dat*)) = *false*
end *Opdracht*

data module *GroepIdKjTable*
begin
imports

Tables
{*Keys* **bound by** [*KEY* → *GroepId*, *eq* → *eq*] **to** *GroepId*
Items **bound by** [*ITEM* → *Kj*] **to** *Kj*
renamed by [*TABLE* → *GroepIdKjTable*, *empty-table* → *leeg-GroepIdKjTable*]}
end *GroepIdKjTable*

data module *GroepIdDuurTable*
begin
imports
Tables
{*Keys* **bound by** [*KEY* → *GroepId*, *eq* → *eq*] **to** *GroepId*
Items **bound by** [*ITEM* → *Duur1*] **to** *Tijd*
renamed by [*TABLE* → *GroepIdDuurTable*, *empty-table* → *leeg-GroepIdDuurTable*]}
end *GroepIdDuurTable*

data module *StrategieId*
begin
exports
begin
sorts *StrategieId*
functions
sid : *CHAR-STRING* → *StrategieId*
eq : *StrategieId* # *StrategieId* → *BOOLEAN*
end
imports *Non-Empty-Character-Strings*[64]
variables *cs*, *cs'* : → *CHAR-STRING*
equations
[e1] *eq*(*sid*(*cs*), *sid*(*cs'*)) = *eq*(*cs*, *cs'*)
end *StrategieId*

data module *GroepIds*
begin
imports
QuasiSet[63]
{*Item* **bound by** [*Q-ELEMENT* → *GroepId*, *eq* → *eq*] **to** *GroepId*
renamed by [*QuasiSet* → *GroepIds*, *leeg* → *0gids*]}
end *GroepIds*

data module *LichtpuntIds*
begin
imports
QuasiSet[63]
{*Item* **bound by** [*Q-ELEMENT* → *LichtpuntId*, *eq* → *eq*] **to** *LichtpuntId*
renamed by [*QuasiSet* → *LichtpuntIds*, *leeg* → *0lids*]}
end *LichtpuntIds*

data module *Kosten*
begin
exports
begin
sorts *Kosten*

functions
k : *DIGIT-SEQ* $\rightarrow$ *Kosten*
eq : *Kosten* # *Kosten* $\rightarrow$ *BOOLEAN*
end
imports *Floating-Points-3-4*
variables *d1*, *d2* : $\rightarrow$ *DIGIT-SEQ*
equations
[i3] $k(d1) = k(\textit{cut-left-zeroes}(d1))$ **when** $\textit{first}(d1) = 0$
[eq1] $eq(k(d1), k(d2)) = eq(\textit{cut-left-zeroes}(d1), \textit{cut-left-zeroes}(d2))$
end *Kosten*

data module *BrandKental*
begin
exports
begin
sorts *BrandKental*
functions
bk : *DIGIT-SEQ* $\rightarrow$ *BrandKental*
On : $\rightarrow$ *BrandKental*
Off : $\rightarrow$ *BrandKental*
i : *BrandKental* $\rightarrow$ *FP*
eq : *BrandKental* # *BrandKental* $\rightarrow$ *BOOLEAN*
end
imports *Floating-Points-3-4*
variables *d1*, *d2* : $\rightarrow$ *DIGIT-SEQ*
equations
[i1] $i(bk(d1)) = \textit{make-fp}(\hat{}0, d1)$
[i2] $i(On) = \textit{make-fp}(\hat{}1)$
[i3] $Off = bk(\hat{}0)$
[eq1] $eq(bk(d1), bk(d2)) = eq(\textit{cut-right-zeroes}(d1), \textit{cut-right-zeroes}(d2))$
[eg2] $eq(bk(d1), On) = \textit{false}$
[eg3] $eq(On, bk(d1)) = \textit{false}$
[eg4] $eq(On, On) = \textit{true}$
[eq5] $eq(On, Off) = \textit{false}$
[eq6] $eq(Off, On) = \textit{false}$
[eq7] $eq(Off, Off) = \textit{true}$
[eq8] $eq(bk(d1), Off) = eq(\textit{cut-right-zeroes}(d1), \hat{}0)$
[eq9] $eq(Off, bk(d1)) = eq(\textit{cut-right-zeroes}(d1), \hat{}0)$
end *BrandKental*

data module *Lichtpunt*
begin
exports
begin
sorts *Lichtpunt*
functions
lichtpunt : *LichtpuntId* # *Lamptype* # *LichtpuntLocatie* # *BrandKental* $\rightarrow$ *Lichtpunt*
eq : *Lichtpunt* # *Lichtpunt* $\rightarrow$ *BOOLEAN*
end
imports *BrandKental*[68], *LichtpuntLocatie*[60], *Lamptype*[62], *LichtpuntId*[65]

variables
lid, *lid'* : → *LichtpuntId*
lt, *lt'* : → *Lamptype*
ll, *ll'* : → *LichtpuntLocatie*
bk, *bk'* : → *BrandKental*
equations
[eq1] *eq*(*lichtpunt*(*lid*, *lt*, *ll*, *bk*), *lichtpunt*(*lid'*, *lt'*, *ll'*, *bk'*)) =
and(*eq*(*lid*, *lid'*), *and*(*eq*(*lt*, *lt'*), *and*(*eq*(*ll*, *ll'*), *eq*(*bk*, *bk'*))))
end *Lichtpunt*

data module *Lichtpunten*
begin
exports
begin
functions
lids : *Lichtpunten* → *LichtpuntIds*
end
imports
QuasiSet[63]
{*Item* **bound by** [*Q-ELEMENT* → *Lichtpunt*, *eq* → *eq*] **to** *Lichtpunt*
renamed by [*QuasiSet* → *Lichtpunten*, *leeg* → *0lpn*]},
LichtpuntIds[67]
variables
lid : → *LichtpuntId*
lt : → *Lamptype*
ll : → *LichtpuntLocatie*
bk : → *BrandKental*
lpn : → *Lichtpunten*
equations
[i1] *lids*(*0lpn*) = *0lids*
[l2] *lids*(*lichtpunt*(*lid*, *lt*, *ll*, *bk*) ∼ *lpn*) = *lid* ∼ *lids*(*lpn*)
end *Lichtpunten*

data module *Groep*
begin
exports
begin
sorts *Groep*
functions
groep : *GroepId* # *Lichtpunten* # *Kosten* # *Kosten* # *Kosten* # *BrandKental* → *Groep*
gid : *Groep* → *GroepId*
lpn : *Groep* → *Lichtpunten*
lids : *Groep* → *LichtpuntIds*
a : *Groep* → *Kosten*
c : *Groep* → *Kosten*
p : *Groep* → *Kosten*
b : *Groep* → *BrandKental*
eq : *Groep* # *Groep* → *BOOLEAN*
end
imports *Lichtpunten*[69], *Kosten*[67], *GroepId*[65]
variables

gid, gid' : → *GroepId*
lpn, lpn' : → *Lichtpunten*
k1, k1', k2, k2', k3, k3' : → *Kosten*
bk, bk' : → *BrandKental*
g1 : → *Groep*

equations

[p1] $gid(groep(gid, lpn, k1, k2, k3, bk)) = gid$
[p2] $lpn(groep(gid, lpn, k1, k2, k3, bk)) = lpn$
[p3] $lids(g1) = lids(lpn(g1))$
[p4] $a(groep(gid, lpn, k1, k2, k3, bk)) = k1$
[p5] $c(groep(gid, lpn, k1, k2, k3, bk)) = k2$
[p6] $p(groep(gid, lpn, k1, k2, k3, bk)) = k3$
[p7] $b(groep(gid, lpn, k1, k2, k3, bk)) = bk$
[eq1] $eq(groep(gid, lpn, k1, k2, k3, bk), groep(gid', lpn', k1', k2', k3', bk')) =$
$and(eq(gid, gid'),$
$and(eq(lpn, lpn'), and(eq(k1, k1'), and(eq(k2, k2'), and(eq(k3, k3'), eq(bk, bk'))))))$

end *Groep*

data module *Groepen*
begin
exports
begin
functions
gids : *Groepen* → *GroepIds*
lpn : *Groepen* → *Lichtpunten*
end
imports
QuasiSet[63]
{*Item* **bound by** [*Q-ELEMENT* → *Groep*, *eq* → *eq*] **to** *Groep*
renamed by [*QuasiSet* → *Groepen*, *leeg* → *0gpn*]},
GroepIds[67]
variables
g1 : → *Groep*
gn : → *Groepen*
equations
[g1] $gids(0gpn) = 0gids$
[g2] $gids(g1 \sim gn) = gid(g1) \sim gids(gn)$
[l1] $lpn(0gpn) = 0lpn$
[l2] $lpn(g1 \sim gn) = lpn(g1) \sim lpn(gn)$

end *Groepen*

data module *Configuratie*
begin
exports
begin
sorts *Configuratie*
functions
configuratie : *Groepen # Kosten # StrategieId* → *Configuratie*
groepen : *Configuratie* → *Groepen*
basiskosten : *Configuratie* → *Kosten*
sid : *Configuratie* → *StrategieId*

vorm-eis : *Configuratie* → *BOOLEAN*
eq : *Configuratie # Configuratie* → *BOOLEAN*

end
imports *Groepen*[70], *StrategieId*[67]
functions
ve': *Groepen # GroepIds # LichtpuntIds* → *BOOLEAN*
variables
sid, *sid'* : → *StrategieId*
gids : → *GroepIds*
lids : → *LichtpuntIds*
basiskosten, *basiskosten'* : → *Kosten*
g1 : → *Groep*
gn, *gn'* : → *Groepen*
cf : → *Configuratie*
equations
[g1] *groepen*(*configuratie*(*gn*, *basiskosten*, *sid*)) = *gn*
[g2] *basiskosten*(*configuratie*(*gn*, *basiskosten*, *sid*)) = *basiskosten*
[g3] *sid*(*configuratie*(*gn*, *basiskosten*, *sid*)) = *sid*
[eq1] *eq*(*configuratie*(*gn*, *basiskosten*, *sid*), *configuratie*(*gn'*, *basiskosten'*, *sid'*)) =
and(*eq*(*gn*, *gn'*), *and*(*eq*(*basiskosten*, *basiskosten'*), *eq*(*sid*, *sid'*)))
[ve1] *vorm-eis*(*cf*) = *false* **when** *eq*(*groepen*(*cf*), *0gpn*) = *true*
[ve2] *vorm-eis*(*cf*) = *ve'*(*groepen*(*cf*), *0gids*, *0lids*) **when** *eq*(*groepen*(*cf*), *0gpn*) = *false*
[ve3] *ve'*(*0gpn*, *gids*, *lids*) = *true*
[ve4] *ve'*(*g1* ∼ *gn*, *gids*, *lids*) =
and(*ve'*(*gn*, *gid*(*g1*) ∼ *gids*, *lids*(*g1*) ∼ *lids*),
and(*not*(*element*(*gid*(*g1*), *gids*)),
and(*eq*(*isect*(*lids*(*g1*), *lids*), *0lids*), *not*(*eq*(*lids*(*g1*), *0lids*)))))
end *Configuratie*

data module *Vli*
begin
exports
begin
sorts *Vli*
functions
vli : *VliId # Configuratie # Datum* → *Vli*
vid : *Vli* → *VliId*
configuratie : *Vli* → *Configuratie*
tijd : *Vli* → *Datum*
lids : *Vli* → *LichtpuntIds*
sid : *Vli* → *StrategieId*
groep : *GroepId # Vli* → *Groepen*
eq : *Vli # Vli* → *BOOLEAN*
vorm-eis : *Vli* → *BOOLEAN*
end
imports *VliId*[65], *Configuratie*[70], *Tijd*[61]
functions
groep : *GroepId # Groepen* → *Groepen*
variables
gid : → *GroepId*

vid : → *VliId*
dat : → *Datum*
g1 : → *Groep*
gn : → *Groepen*
cf : → *Configuratie*
vli, *vli'* : → *Vli*

equations

[v1] *vid*(*vli*(*vid*, *cf*, *dat*)) = *vid*
[c1] *configuratie*(*vli*(*vid*, *cf*, *dat*)) = *cf*
[t1] *tijd*(*vli*(*vid*, *cf*, *dat*)) = *dat*
[l1] *lids*(*vli*) = *lids*(*lpn*(*groepen*(*configuratie*(*vli*))))
[st1] *sid*(*vli*) = *sid*(*configuratie*(*vli*))
[g1] *groep*(*gid*, *vli*) = *groep*(*gid*, *groepen*(*configuratie*(*vli*)))
[g2] *groep*(*gid*, *0gpn*) = *0gpn*
[g3] *groep*(*gid*, *g1* ∼ *gn*) = *g1* ∼ *0gpn* **when** *eq*(*gid*, *gid*(*g1*)) = *true*
[g4] *groep*(*gid*, *g1* ∼ *gn*) = *groep*(*gid*, *gn*) **when** *eq*(*gid*, *gid*(*g1*)) = *false*
[eq1] *eq*(*vli*, *vli'*) =
and(*eq*(*vid*(*vli*), *vid*(*vli'*)),
and(*eq*(*configuratie*(*vli*), *configuratie*(*vli'*)), *eq*(*tijd*(*vli*), *tijd*(*vli'*))))
[ve1] *vorm-eis*(*vli*) = *vorm-eis*(*configuratie*(*vli*))

end *Vli*

data module *Schema*

begin

exports

begin

sorts *Schema*

functions

schema-IHG : *Duur1* # *GroepIdKjTable* → *Schema*
schema-OBR : *GroepIdDuurTable* → *Schema*
duur : *Schema* → *Duur1*
tabel : *Schema* → *GroepIdDuurTable*
schema : *Vli* → *Schema*
schema : *Configuratie* → *Schema*

end

imports *Vli*[71], *GroepIdKjTable*[66], *GroepIdDuurTable*[67], *Parameters*[84]

variables

dr : → *Duur1*
gkjTabel : → *GroepIdKjTable*
gdrTabel : → *GroepIdDuurTable*
vli : → *Vli*

equations

[d1] *duur*(*schema-IHG*(*dr*, *gkjTabel*)) = *dr*
[d2] *duur*(*schema-OBR*(*gdrTabel*)) = *een*
[t1] *tabel*(*schema-IHG*(*dr*, *gkjTabel*)) = *leeg-GroepIdDuurTable*
[t2] *tabel*(*schema-OBR*(*gdrTabel*)) = *gdrTabel*
[s1] *schema*(*vli*) = *schema*(*configuratie*(*vli*))

end *Schema*

data module *Betrouwbaarheid*

begin
  exports
  begin
    sorts *Betrouwbaarheid*
    functions
      betrouwbaarheid : *FP # Duur1* → *Betrouwbaarheid*
      betrouwbaarheid : *Configuratie* → *Betrouwbaarheid*
  end
  imports *Schema*[72]
end *Betrouwbaarheid*

data module *Exploitatie*
begin
  exports
  begin
    sorts *Exploitatie*
    functions
      exploitatie : *Kosten # Duur1* → *Exploitatie*
      exploitatie : *Configuratie* → *Exploitatie*
  end
  imports *Schema*[72]
end *Exploitatie*

data module *Vlis*
begin
  exports
  begin
    functions
      vids : *Vlis* → *VliIds*
  end
  imports
    List[60]
      {*Item* **bound by** [*L-ELEMENT* → *Vli*, *eq* → *eq*] **to** *Vli*
       renamed by [*List* → *Vlis*, *leeg* → *0vlis*]},
    VliIds[65]
  variables
    vli : → *Vli*
    vlis : → *Vlis*
  equations
  [v1] *vids*(*0vlis*) = *0vids*
  [v2] *vids*(*vli* ∼ *vlis*) = *vid*(*vli*) ∼ *vids*(*vlis*)
end *Vlis*

15.2 Datastructuren van `Database`

Hieronder volgen specificaties van datastructuren wier objecten door het proces `Database`, zie Paragraaf 15.5, beheerd en bewerkt worden. Er wordt voort gebouwd op de structuren uit de vorige paragraaf. De structuren in deze paragraaf zijn naar onze smaak niet af, er zijn nog veel vragen en onduidelijkheden op te lossen.

In de module `Gebeurtenissen` worden enkele functies gedefinieerd die dienen om vragen te kunnen stellen over de opgeslagen gegevens, deze functies worden veel gebruikt door de processen in keuzepunten. Voorbeelden van deze functies zijn de al eerder genoemde functies `inmarge` en `onder-schema`.

data module *Opdrachten*
begin
exports
begin
functions
verwijder-opdrachten : *VliId # Datum # Opdrachten* → *Opdrachten*
end
imports
List[60]
{*Item* **bound by** [*L-ELEMENT* → *Opdracht*] **to** *Opdracht*
renamed by [*List* → *Opdrachten*, *leeg* → *0opn*]},
Schema[72]
variables
dat, *dat'* : → *Datum*
lid, *lid'* : → *LichtpuntId*
gid, *gid'* : → *GroepId*
vid, *vid'* : → *VliId*
op : → *Opdracht*
opn : → *Opdrachten*
equations
[r1] *verwijder-opdrachten*(*vid*, *dat*, *0opn*) = *0opn*
[r2] *verwijder-opdrachten*(*vid*, *dat*, *op* ∼ *opn*) = *verwijder-opdrachten*(*vid*, *dat*, *opn*)
when *and*(*eq*(*vid*, *vid*(*op*)), *tijd*(*op*) ≥ *dat*) = *true*
[r3] *verwijder-opdrachten*(*vid*, *dat*, *op* ∼ *opn*) = *op* ∼ *verwijder-opdrachten*(*vid*, *dat*, *opn*)
when *and*(*eq*(*vid*, *vid*(*op*)), *tijd*(*op*) ≥ *dat*) = *false*
[o1] *correctief*(*vid*, *lid*, *dat*) ∼ (*correctief*(*vid'*, *lid'*, *dat'*) ∼ *opn*) =
correctief(*vid'*, *lid'*, *dat'*) ∼ (*correctief*(*vid*, *lid*, *dat*) ∼ *opn*)
when *vid* > *vid'* = *true*
[o2] *correctief*(*vid*, *lid*, *dat*) ∼ (*preventief*(*vid'*, *gid*, *dat'*) ∼ *opn*) =
preventief(*vid'*, *gid*, *dat'*) ∼ (*correctief*(*vid*, *lid*, *dat*) ∼ *opn*)
when *vid* > *vid'* = *true*
[o3] *preventief*(*vid'*, *gid'*, *dat'*) ∼ (*correctief*(*vid*, *lid*, *dat*) ∼ *opn*) =
correctief(*vid*, *lid*, *dat*) ∼ (*preventief*(*vid'*, *gid*, *dat'*) ∼ *opn*)
when *vid'* > *vid* = *true*
[o4] *preventief*(*vid*, *gid*, *dat*) ∼ (*preventief*(*vid'*, *gid'*, *dat'*) ∼ *opn*) =
preventief(*vid'*, *gid'*, *dat'*) ∼ (*preventief*(*vid*, *gid*, *dat*) ∼ *opn*)
when *vid* > *vid'* = *true*
[o5] *preventief*(*vid*, *gid*, *dat'*) ∼ (*correctief*(*vid*, *lid*, *dat*) ∼ *opn*) =
correctief(*vid*, *lid*, *dat*) ∼ (*preventief*(*vid*, *gid*, *dat'*) ∼ *opn*)
[o6] *correctief*(*vid*, *lid*, *dat*) ∼ (*correctief*(*vid*, *lid'*, *dat'*) ∼ *opn*) =
correctief(*vid*, *lid'*, *dat'*) ∼ (*correctief*(*vid*, *lid*, *dat*) ∼ *opn*)
when *lid* > *lid'* = *true*
[o7] *preventief*(*vid*, *gid*, *dat*) ∼ (*preventief*(*vid*, *gid'*, *dat'*) ∼ *opn*) =
preventief(*vid*, *gid'*, *dat'*) ∼ (*preventief*(*vid*, *gid*, *dat*) ∼ *opn*)
when *gid* > *gid'* = *true*
[o8] *correctief*(*vid*, *lid*, *dat*) ∼ (*correctief*(*vid*, *lid*, *dat'*) ∼ *opn*) =
correctief(*vid*, *lid*, *min*(*dat*, *dat'*)) ∼ *opn*
[o9] *preventief*(*vid*, *gid*, *dat*) ∼ (*preventief*(*vid*, *gid*, *dat*) ∼ *opn*) = *preventief*(*vid*, *gid*, *dat*) ∼ *opn*
[o10] *preventief*(*vid*, *gid*, *dat*) ∼ (*preventief*(*vid*, *gid*, *dat'*) ∼ *opn*) =
preventief(*vid*, *gid*, *dat'*) ∼ (*preventief*(*vid*, *gid*, *dat*) ∼ *opn*)
when *dat'* < *dat* = *true*

end *Opdrachten*

data module *Bericht*
begin
exports
begin
sorts *Bericht*
functions

triviaal-bericht	:	→ *Bericht*
geef-vli	: *VliId*	→ *Bericht*
start	: *Vli*	→ *Bericht*
tijd	: *Datum*	→ *Bericht*
analyse-nieuwe-configuratie	:	→ *Bericht*
analyse-configuratie	: *VliId*	→ *Bericht*
bekend	: *VliId*	→ *Bericht*
onbekend	: *VliId*	→ *Bericht*
stop	:	→ *Bericht*
ga-door	:	→ *Bericht*
oke	:	→ *Bericht*
niet-oke	:	→ *Bericht*
defect	: *VliId # LichtpuntIds # Datum*	→ *Bericht*
uitgevoerd	: *Opdracht # Datum*	→ *Bericht*
stop	: *VliId # Datum*	→ *Bericht*
onderdeel-van	: *LichtpuntIds # VliId*	→ *Bericht*
onder-schema	: *VliId # Datum*	→ *Bericht*
tijd-laatst-uitgevoerde-opdracht	: *VliId*	→ *Bericht*
geen-onderdeelvan	: *LichtpuntIds # VliId*	→ *Bericht*
vli-bestaat-niet	:	→ *Bericht*
editor-inhoud-niet-consistent	:	→ *Bericht*
tijdstip-verdacht	: *Datum*	→ *Bericht*
opdracht-onbekend	: *Opdracht*	→ *Bericht*
starttijd-schema-in-het-verleden	:	→ *Bericht*
Starttijdstip-voor-Stoptijdstip	:	→ *Bericht*
Geen-stop-van-huidig-schema-opgegeven	:	→ *Bericht*
stoptijdstip-conflict-actie	:	→ *Bericht*
stoptijdstip-conflict-start	:	→ *Bericht*
onbekend	: *LichtpuntIds*	→ *Bericht*
niet-onder-schema-op-tijdstip	: *VliId # Datum*	→ *Bericht*
defect-gedetecteerd-in-toekomst	:	→ *Bericht*
configuratie-inconsistent	:	→ *Bericht*
geen-schema-gevonden	:	→ *Bericht*
verwijder-opdrachten	: *VliId # Datum*	→ *Bericht*
vli-is-reeds-gestopt	: *VliId*	→ *Bericht*
Planning-update-te-laat	:	→ *Bericht*
Opdracht-mogelijk-geantidateerd	:	→ *Bericht*
bepaal-actie	: *VliId # LichtpuntIds # Datum*	→ *Bericht*
bepaal-preventieve-opdrachten	: *Datum # Datum*	→ *Bericht*
bepaal-preventieve-opdrachten	: *Datum # Datum # Vli*	→ *Bericht*
preventieve-remplace-alle-groepen	: *Vli*	→ *Bericht*
start-of-stop	: *VliId*	→ *Bericht*

soort-bericht : *Bericht* → *NATURAL*
eq : *Bericht # Bericht* → *BOOLEAN*

end

imports *Vli*[71], *Opdracht*[66]

variables

dat, *dat'*, *dat1*, *dat2*, *dat1'*, *dat2'* : → *Datum*
vid, *vid'* : → *VliId*
lids, *lids'* : → *LichtpuntIds*
op, *op'* : → *Opdracht*
vli, *vli'* : → *Vli*
br, *br'* : → *Bericht*

equations

[sb1] *soort-bericht*(*start*(*vli*)) = *nat*(^1)
[sb2] *soort-bericht*(*defect*(*vid*, *lids*, *dat*)) = *nat*(^2)
[sb3] *soort-bericht*(*uitgevoerd*(*op*, *dat*)) = *nat*(^3)
[sb4] *soort-bericht*(*stop*(*vid*, *dat*)) = *nat*(^4)
[sb5] *soort-bericht*(*geef-vli*(*vid*)) = *nat*(^5)
[sb6] *soort-bericht*(*tijd*(*dat*)) = *nat*(^6)
[sb7] *soort-bericht*(*analyse-nieuwe-configuratie*) = *nat*(^7)
[sb8] *soort-bericht*(*analyse-configuratie*(*vid*)) = *nat*(^8)
[sb9] *soort-bericht*(*bekend*(*vid*)) = *nat*(^9)
[sb10] *soort-bericht*(*stop*) = *nat*(^1 ^ 0)
[sb11] *soort-bericht*(*ga-door*) = *nat*(^1 ^ 1)
[sb12] *soort-bericht*(*oke*) = *nat*(^1 ^ 2)
[sb13] *soort-bericht*(*niet-oke*) = *nat*(^1 ^ 3)
[sb14] *soort-bericht*(*onderdeel-van*(*lids*, *vid*)) = *nat*(^1 ^ 4)
[sb15] *soort-bericht*(*onder-schema*(*vid*, *dat*)) = *nat*(^1 ^ 5)
[sb16] *soort-bericht*(*tijd-laatst-uitgevoerde-opdracht*(*vid*)) = *nat*(^1 ^ 6)
[sb17] *soort-bericht*(*geen-onderdeelvan*(*lids*, *vid*)) = *nat*(^1 ^ 7)
[sb18] *soort-bericht*(*vli-bestaat-niet*) = *nat*(^1 ^ 8)
[sb19] *soort-bericht*(*editor-inhoud-niet-consistent*) = *nat*(^1 ^ 9)
[sb20] *soort-bericht*(*tijdstip-verdacht*(*dat*)) = *nat*(^2 ^ 0)
[sb21] *soort-bericht*(*opdracht-onbekend*(*op*)) = *nat*(^2 ^ 1)
[sb22] *soort-bericht*(*starttijd-schema-in-het-verleden*) = *nat*(^2 ^ 2)
[sb23] *soort-bericht*(*Starttijdstip-voor-Stoptijdstip*) = *nat*(^2 ^ 3)
[sb24] *soort-bericht*(*Geen-stop-van-huidig-schema-opgegeven*) = *nat*(^2 ^ 4)
[sb25] *soort-bericht*(*stoptijdstip-conflict-actie*) = *nat*(^2 ^ 5)
[sb26] *soort-bericht*(*stoptijdstip-conflict-start*) = *nat*(^2 ^ 6)
[sb27] *soort-bericht*(*onbekend*(*lids*)) = *nat*(^2 ^ 7)
[sb28] *soort-bericht*(*niet-onder-schema-op-tijdstip*(*vid*, *dat*)) = *nat*(^2 ^ 8)
[sb29] *soort-bericht*(*defect-gedetecteerd-in-toekomst*) = *nat*(^2 ^ 9)
[sb30] *soort-bericht*(*onbekend*(*vid*)) = *nat*(^3 ^ 0)
[sb31] *soort-bericht*(*configuratie-inconsistent*) = *nat*(^3 ^ 1)
[sb32] *soort-bericht*(*geen-schema-gevonden*) = *nat*(^3 ^ 2)
[sb33] *soort-bericht*(*verwijder-opdrachten*(*vid*, *dat*)) = *nat*(^3 ^ 3)
[sb34] *soort-bericht*(*start-of-stop*(*vid*)) = *nat*(^3 ^ 4)
[sb35] *soort-bericht*(*vli-is-reeds-gestopt*(*vid*)) = *nat*(^3 ^ 5)
[sb36] *soort-bericht*(*Planning-update-te-laat*) = *nat*(^3 ^ 6)
[sb37] *soort-bericht*(*Opdracht-mogelijk-geantidateerd*) = *nat*(^3 ^ 7)
[sb38] *soort-bericht*(*bepaal-actie*(*vid*, *lids*, *dat*)) = *nat*(^3 ^ 8)

[sb39] *soort-bericht*(*bepaal-preventieve-opdrachten*(*dat1*, *dat2*)) = *nat*(^3 ^ 9)
[sb40] *soort-bericht*(*bepaal-preventieve-opdrachten*(*dat1*, *dat2*, *vli*)) = *nat*(^4 ^ 0)
[sb41] *soort-bericht*(*preventieve-remplace-alle-groepen*(*vli*)) = *nat*(^4 ^ 1)
[sb42] *soort-bericht*(*triviaal-bericht*) = *nat*(^4 ^ 2)
[eq1] *eq*(*start*(*vli*), *start*(*vli'*)) = *eq*(*vid*(*vli*), *vid*(*vli'*))
[eq2] *eq*(*defect*(*vid*, *lids*, *dat*), *defect*(*vid'*, *lids'*, *dat'*)) =
and(*eq*(*vid*, *vid'*), *and*(*eq*(*lids*, *lids'*), *eq*(*dat*, *dat'*)))
[eq3] *eq*(*uitgevoerd*(*op*, *dat*), *uitgevoerd*(*op'*, *dat'*)) = *and*(*eq*(*op*, *op'*), *eq*(*dat*, *dat'*))
[eq4] *eq*(*stop*(*vid*, *dat*), *stop*(*vid'*, *dat'*)) = *and*(*eq*(*vid*, *vid'*), *eq*(*dat*, *dat'*))
[eq5] *eq*(*geef-vli*(*vid*), *geef-vli*(*vid'*)) = *eq*(*vid*, *vid'*)
[eq6] *eq*(*tijd*(*dat*), *tijd*(*dat'*)) = *eq*(*dat*, *dat'*)
[eq7] *eq*(*analyse-nieuwe-configuratie*, *analyse-nieuwe-configuratie*) = *true*
[eq8] *eq*(*analyse-configuratie*(*vid*), *analyse-configuratie*(*vid'*)) = *eq*(*vid*, *vid'*)
[eq9] *eq*(*bekend*(*vid*), *bekend*(*vid'*)) = *eq*(*vid*, *vid'*)
[eq10] *eq*(*stop*, *stop*) = *true*
[eq11] *eq*(*ga-door*, *ga-door*) = *true*
[eq12] *eq*(*oke*, *oke*) = *true*
[eq13] *eq*(*niet-oke*, *niet-oke*) = *true*
[eq14] *eq*(*onderdeel-van*(*lids*, *vid*), *onderdeel-van*(*lids'*, *vid'*)) = *and*(*eq*(*lids*, *lids'*), *eq*(*vid*, *vid'*))
[eq15] *eq*(*onder-schema*(*vid*, *dat*), *onder-schema*(*vid'*, *dat'*)) = *and*(*eq*(*vid*, *vid'*), *eq*(*dat*, *dat'*))
[eq16] *eq*(*tijd-laatst-uitgevoerde-opdracht*(*vid*), *tijd-laatst-uitgevoerde-opdracht*(*vid'*)) =
eq(*vid*, *vid'*)
[eq17] *eq*(*geen-onderdeelvan*(*lids*, *vid*), *geen-onderdeelvan*(*lids*, *vid*)) =
and(*eq*(*lids*, *lids'*), *eq*(*vid*, *vid'*))
[eq18] *eq*(*vli-bestaat-niet*, *vli-bestaat-niet*) = *true*
[eq19] *eq*(*editor-inhoud-niet-consistent*, *editor-inhoud-niet-consistent*) = *true*
[eq20] *eq*(*tijdstip-verdacht*(*dat*), *tijdstip-verdacht*(*dat'*)) = *eq*(*dat*, *dat'*)
[eq21] *eq*(*opdracht-onbekend*(*op*), *opdracht-onbekend*(*op'*)) = *eq*(*op*, *op'*)
[eq22] *eq*(*starttijd-schema-in-het-verleden*, *starttijd-schema-in-het-verleden*) = *true*
[eq23] *eq*(*Starttijdstip-voor-Stoptijdstip*, *Starttijdstip-voor-Stoptijdstip*) = *true*
[eq24] *eq*(*Geen-stop-van-huidig-schema-opgegeven*, *Geen-stop-van-huidig-schema-opgegeven*) =
true
[eq25] *eq*(*stoptijdstip-conflict-actie*, *stoptijdstip-conflict-actie*) = *true*
[eq26] *eq*(*stoptijdstip-conflict-start*, *stoptijdstip-conflict-start*) = *true*
[eq27] *eq*(*onbekend*(*lids*), *onbekend*(*lids'*)) = *eq*(*lids*, *lids'*)
[eq28] *eq*(*niet-onder-schema-op-tijdstip*(*vid*, *dat*), *niet-onder-schema-op-tijdstip*(*vid'*, *dat'*)) =
and(*eq*(*vid*, *vid'*), *eq*(*dat*, *dat'*))
[eq29] *eq*(*defect-gedetecteerd-in-toekomst*, *defect-gedetecteerd-in-toekomst*) = *true*
[eq30] *eq*(*onbekend*(*vid*), *onbekend*(*vid'*)) = *eq*(*vid*, *vid'*)
[eq31] *eq*(*configuratie-inconsistent*, *configuratie-inconsistent*) = *true*
[eq32] *eq*(*geen-schema-gevonden*, *geen-schema-gevonden*) = *true*
[eq33] *eq*(*verwijder-opdrachten*(*vid*, *dat*), *verwijder-opdrachten*(*vid'*, *dat'*)) =
and(*eq*(*vid*, *vid'*), *eq*(*dat*, *dat'*))
[eq34] *eq*(*start-of-stop*(*vid*), *start-of-stop*(*vid'*)) = *eq*(*vid*, *vid'*)
[eq35] *eq*(*vli-is-reeds-gestopt*(*vid*), *vli-is-reeds-gestopt*(*vid'*)) = *eq*(*vid*, *vid'*)
[eq36] *eq*(*Planning-update-te-laat*, *Planning-update-te-laat*) = *true*
[eq37] *eq*(*Opdracht-mogelijk-geantidateerd*, *Opdracht-mogelijk-geantidateerd*) = *true*
[eq38] *eq*(*bepaal-actie*(*vid*, *lids*, *dat*), *bepaal-actie*(*vid'*, *lids'*, *dat'*)) =
and(*eq*(*vid*, *vid'*), *and*(*eq*(*lids*, *lids'*), *eq*(*dat*, *dat'*)))
[eq39] *eq*(*bepaal-preventieve-opdrachten*(*dat1*, *dat2*), *bepaal-preventieve-opdrachten*(*dat1*, *dat2*)) =
and(*eq*(*dat1*, *dat1'*), *eq*(*dat2*, *dat2'*))

[eq40] *eq*(*bepaal-preventieve-opdrachten*(*dat1*, *dat2*, *vli*),
bepaal-preventieve-opdrachten(*dat1'*, *dat2'*, *vli'*))
= *and*(*eq*(*dat1*, *dat1'*), *and*(*eq*(*dat2*, *dat2'*), *eq*(*vli*, *vli'*)))
[eq41] *eq*(*preventieve-remplace-alle-groepen*(*vli*), *preventieve-remplace-alle-groepen*(*vli'*)) =
eq(*vli*, *vli'*)
[eq42] *eq*(*triviaal-bericht*, *triviaal-bericht*) = *true*
[eq43] *eq*(*br*, *br'*) = *false* **when** *eq*(*soort-bericht*(*br*), *soort-bericht*(*br'*)) = *false*

end *Bericht*

data module *Gebeurtenis*
begin
exports
begin
sorts *Gebeurtenis*
functions
gb : *Bericht* → *Gebeurtenis*
eq : *Gebeurtenis* # *Gebeurtenis* → *BOOLEAN*
end
imports *Bericht*[75]
variables *br*, *br'* : → *Bericht*
equations
[e1] *eq*(*gb*(*br*), *gb*(*br'*)) = *eq*(*br*, *br'*)
end *Gebeurtenis*

data module *Gebeurtenissen*
begin
exports
begin
functions
vli : *VliId* # *Gebeurtenissen* → *Vlis*
bekend : *VliId* # *Gebeurtenissen* → *BOOLEAN*
onderdeel-van : *LichtpuntIds* # *VliId* # *Gebeurtenissen* → *BOOLEAN*
start-of-stop : *VliId* # *Gebeurtenissen* → *Bericht*
inmarge : *Datum* # *Opdracht* # *Gebeurtenissen* → *BOOLEAN*
verwijder-eerst-volgende-start : *VliId* # *Gebeurtenissen* → *Gebeurtenissen*
onder-schema : *VliId* # *Datum* # *Gebeurtenissen* → *BOOLEAN*
end
imports
List[60]
{*Item* **bound by** [*L-ELEMENT* → *Gebeurtenis*] **to** *Gebeurtenis*
renamed by [*List* → *Gebeurtenissen*, *leeg* → *0gbs*]},
Vlis[73], *Parameters*[84]
functions
onderdeel-van-1 : *LichtpuntIds* # *Vlis* → *BOOLEAN*
onder-schema-1 : *Vlis* # *Datum* # *Datum* # *Gebeurtenissen* → *BOOLEAN*
variables
vid, *vid'* : → *VliId*
lids : → *LichtpuntIds*
dat, *dat'* : → *Datum*
op : → *Opdracht*
vli : → *Vli*

vlis $: \rightarrow$ *Vlis*
br $: \rightarrow$ *Bericht*
gbs $: \rightarrow$ *Gebeurtenissen*

equations

[gb1] $gb(triviaal\text{-}bericht) \sim gbs = gbs$
[vl1] $vli(vid, 0gbs) = 0vlis$
[vl2] $vli(vid, gb(start(vli)) \sim gbs) = vli \sim 0vlis$ **when** $eq(vid, vid(vli)) = true$
[vl3] $vli(vid, gb(start(vli)) \sim gbs) = vli(vid, gbs)$ **when** $eq(vid, vid(vli)) = false$
[vl4] $vli(vid, gb(br) \sim gbs) = vli(vid, gbs)$ **when** $eq(soort\text{-}bericht(br), nat(\hat{}1)) = false$
[b1] $bekend(vid, gbs) = not(eq(vli(vid, gbs), 0vlis))$
[o1] $onderdeel\text{-}van(lids, vid, gbs) = onderdeel\text{-}van\text{-}1(lids, vli(vid, gbs))$
[o2] $onderdeel\text{-}van\text{-}1(lids, 0vlis) = false$
[o3] $onderdeel\text{-}van\text{-}1(lids, vli \sim vlis) = subset(lids, lids(vli))$
[ss1] $start\text{-}of\text{-}stop(vid, 0gbs) = triviaal\text{-}bericht$
[ss2] $start\text{-}of\text{-}stop(vid, gb(stop(vid, dat)) \sim gbs) = stop(vid, dat)$
[ss3] $start\text{-}of\text{-}stop(vid, gb(stop(vid', dat)) \sim gbs) = start\text{-}of\text{-}stop(vid, gbs)$
when $eq(vid, vid') = false$
[ss4] $start\text{-}of\text{-}stop(vid, gb(start(vli)) \sim gbs) = start(vli)$ **when** $eq(vid, vid(vli)) = true$
[ss5] $start\text{-}of\text{-}stop(vid, gb(start(vli)) \sim gbs) = start\text{-}of\text{-}stop(vid, gbs)$
when $eq(vid, vid(vli)) = false$
[ss6] $start\text{-}of\text{-}stop(vid, gb(br) \sim gbs) = start\text{-}of\text{-}stop(vid, gbs)$
when $or(eq(soort\text{-}bericht(br), nat(\hat{}1)), eq(soort\text{-}bericht(br), nat(\hat{}4))) = false$
[im1] $inmarge(dat, op, gbs) = eq(dat, tijd(op))$
[vv1] $verwijder\text{-}eerst\text{-}volgende\text{-}start(vid, 0gbs) = 0gbs$
[vv2] $verwijder\text{-}eerst\text{-}volgende\text{-}start(vid, gb(start(vli)) \sim gbs) = gbs$
when $eq(vid, vid(vli)) = true$
[vv3] $verwijder\text{-}eerst\text{-}volgende\text{-}start(vid, gb(start(vli)) \sim gbs) =$
$gb(start(vli)) \sim verwijder\text{-}eerst\text{-}volgende\text{-}start(vid, gbs)$
when $eq(vid, vid(vli)) = false$
[vv4] $verwijder\text{-}eerst\text{-}volgende\text{-}start(vid, gb(br) \sim gbs) =$
$gb(br) \sim verwijder\text{-}eerst\text{-}volgende\text{-}start(vid, gbs)$
when $eq(soort\text{-}bericht(br), nat(\hat{}1)) = false$
[os1] $onder\text{-}schema(vid, dat, 0gbs) = false$
[os2] $onder\text{-}schema(vid, dat, gb(stop(vid, dat')) \sim gbs) =$
$onder\text{-}schema\text{-}1(vli(vid, gbs), dat, dat', gbs)$
[os3] $onder\text{-}schema(vid, dat, gb(stop(vid', dat')) \sim gbs) = onder\text{-}schema(vid, dat, gbs)$
when $eq(vid, vid') = false$
[os4] $onder\text{-}schema(vid, dat, gb(start(vli)) \sim gbs) =$
$or(and(eq(vid, vid(vli)), tijd(vli) \leq dat), onder\text{-}schema(vid, dat, gbs))$
[os5] $onder\text{-}schema(vid, dat, gb(br) \sim gbs) = onder\text{-}schema(vid, dat, gbs)$
when $or(eq(soort\text{-}bericht(br), nat(\hat{}1)), eq(soort\text{-}bericht(br), nat(\hat{}4))) = false$
[os6] $onder\text{-}schema\text{-}1(0vlis, dat, dat', gbs) = false$
[os7] $onder\text{-}schema\text{-}1(vli \sim vlis, dat, dat', gbs) =$
$or(and(tijd(vli) \leq dat, dat < dat'),$
$onder\text{-}schema(vid(vli), dat, verwijder\text{-}eerst\text{-}volgende\text{-}start(vid(vli), gbs)))$

end *Gebeurtenissen*

15.3 LRdata

Deze paragraaf bevat één module, `LRdata`. Deze module is de topmodule van de dataspecificatie, dus afgezien van de modulen die voor simulatie doeleinden zijn toegevoegd. De belangrijkste functies die het BOS berekent zijn hier gespecificeerd, te weten, `bepaal-actie` en `preventieve-opdrachten`. De

eerste functie berekent gegeven een vli, een defect, en een datum wat de te ondernemen remplaceactie moet zijn. De tweede functie berekent gegeven een tijdsspanne de preventieve vervangingsopdrachten voor de vli's die in die tijdsspanne geheel of gedeeltelijk in beheer zijn.

data module *LRdata*
begin
exports
begin
functions

tijd-laatst-uitgevoerde-preventieve-opdracht	:	*VliId # GroepId # Gebeurtenissen*	→ *Datum*
tijd-laatst-uitgevoerde-opdracht	:	*VliId # Gebeurtenissen*	→ *Datum*
preventieve-opdrachten	:	*Datum # Datum # Gebeurtenissen*	→ *Opdrachten*
preventieve-opdrachten	:	*Datum # Datum # Vli*	→ *Opdrachten*
preventief-alle-groepen	:	*Vli*	→ *Opdrachten*
bepaal-actie	:	*VliId # LichtpuntIds # Datum # Gebeurtenissen*	→ *Opdrachten*
merge-opdrachten	:	*Opdrachten # Opdrachten*	→ *Opdrachten*

end
imports *Gebeurtenissen*[78], *Opdrachten*[74]
functions

preventieve-opdrachten-1	:	*Datum # Datum # Vlis*	→ *Opdrachten*
preventieve-opdrachten-IHG	:	*Datum # Datum # VliId # Datum# Schema*	→ *Opdrachten*
preventieve-opdrachten-IHG-1	:	*Datum # Datum # VliId # Datum # Duur1 # GroepId # Kj # NATURAL*	→ *Opdrachten*
preventief-alle-groepen-1	:	*VliId # GroepIds # Datum*	→ *Opdrachten*
bepaal-actie-0	:	*Vlis # LichtpuntIds # Datum # Gebeurtenissen*	→ *Opdrachten*
bepaal-actie-1	:	*Vli # LichtpuntIds # Datum # Gebeurtenissen*	→ *Opdrachten*
bepaal-actie-2	:	*VliId # LichtpuntIds # Datum # LichtpuntIds*	→ *Opdrachten*
bepaal-actie-OBR	:	*VliId # GroepIdDuurTable # LichtpuntIds # Datum # Groepen # Gebeurtenissen*	→ *Opdrachten*
bepaal-actie-OBR-1	:	*VliId # GroepIdDuurTable # LichtpuntIds # Datum # Groepen # RESULT-OF-LOOKUP # Gebeurtenissen*	→ *Opdrachten*
bepaal-actie-OBR-2	:	*VliId # GroepIdDuurTable # LichtpuntIds # Datum # Groepen # LichtpuntIds # Gebeurtenissen*	→ *Opdrachten*
bepaal-actie-OBR-3	:	*VliId # GroepIdDuurTable # LichtpuntIds # Datum # Groepen # RESULT-OF-LOOKUP # LichtpuntIds # Gebeurtenissen*	→ *Opdrachten*
tijd-laatst-uitgevoerde-preventieve-opdracht-1	:	*Vlis # Datum # GroepId # Gebeurtenissen*	→ *Datum*
tijd-laatst-uitgevoerde-preventieve-opdracht-2	:	*Groepen # Datum # VliId # Gebeurtenissen*	→ *Datum*

tijd-laatst-uitgevoerde-opdracht-1 : *Vlis # Datum # Gebeurtenissen* → *Datum*

variables

tel	: → *NATURAL*
kgid	: → *Kj*
dat, *dat1*, *dat2*, *dat3*	: → *Datum*
dr, *basiscyclus*	: → *Duur1*
k1	: → *Kosten*
lid	: → *LichtpuntId*
gid, *gid'*	: → *GroepId*
gdrTable	: → *GroepIdDuurTable*
gkjTable	: → *GroepIdKjTable*
vid, *vid'*	: → *VliId*
sid	: → *StrategieId*
lids, *lids'*	: → *LichtpuntIds*
gids	: → *GroepIds*
g1	: → *Groep*
gn	: → *Groepen*
vli	: → *Vli*
vlis	: → *Vlis*
op	: → *Opdracht*
opn, *opn'*	: → *Opdrachten*
br	: → *Bericht*
gbs	: → *Gebeurtenissen*

equations

[t1] *tijd-laatst-uitgevoerde-preventieve-opdracht*(*vid*, *gid*, *0gbs*) = *startdatum*

[t2] *tijd-laatst-uitgevoerde-preventieve-opdracht*(*vid*, *gid*, *gb*(*uitgevoerd*(*preventief*(*vid*, *gid*, *dat1*), *dat2*)) ∼ *gbs*)
= *tijd-laatst-uitgevoerde-preventieve-opdracht-1*(*vli*(*vid*, *gbs*), *dat2*, *gid*, *gbs*)

[t3] *tijd-laatst-uitgevoerde-preventieve-opdracht*(*vid*, *gid*, *gb*(*uitgevoerd*(*preventief*(*vid'*, *gid'*, *dat1*), *dat2*)) ∼ *gbs*)
= *tijd-laatst-uitgevoerde-preventieve-opdracht*(*vid*, *gid*, *gbs*)
when *and*(*eq*(*vid*, *vid'*), *eq*(*gid*, *gid'*)) = *false*

[t4] *tijd-laatst-uitgevoerde-preventieve-opdracht*(*vid*, *gid*, *gb*(*uitgevoerd*(*correctief*(*vid'*, *lid*, *dat1*), *dat2*)) ∼ *gbs*)
= *tijd-laatst-uitgevoerde-preventieve-opdracht*(*vid*, *gid*, *gbs*)

[t5] *tijd-laatst-uitgevoerde-preventieve-opdracht*(*vid*, *gid*, *gb*(*br*) ∼ *gbs*) =
tijd-laatst-uitgevoerde-preventieve-opdracht(*vid*, *gid*, *gbs*)
when *eq*(*soort-bericht*(*br*), *nat*(ˆ3)) = *false*

[t6] *tijd-laatst-uitgevoerde-preventieve-opdracht-1*(*0vlis*, *dat*, *gid*, *gbs*) = *startdatum*

[t7] *tijd-laatst-uitgevoerde-preventieve-opdracht-1*(*vli* ∼ *vlis*, *dat*, *gid*, *gbs*) =
tijd-laatst-uitgevoerde-preventieve-opdracht-2(*groep*(*gid*, *vli*), *dat*, *vid*(*vli*), *gbs*)

[t8] *tijd-laatst-uitgevoerde-preventieve-opdracht-2*(*0gpn*, *dat*, *vid*, *gbs*) = *startdatum*

[t9] *tijd-laatst-uitgevoerde-preventieve-opdracht-2*(*g1* ∼ *gn*, *dat*, *vid*, *gbs*) =
max(*dat*, *tijd-laatst-uitgevoerde-preventieve-opdracht*(*vid*, *gid*(*g1*), *gbs*))

[la1] *tijd-laatst-uitgevoerde-opdracht*(*vid*, *0gbs*) = *startdatum*

[la2] *tijd-laatst-uitgevoerde-opdracht*(*vid*, *gb*(*uitgevoerd*(*op*, *dat*)) ∼ *gbs*) =
tijd-laatst-uitgevoerde-opdracht-1(*vli*(*vid*, *gbs*), *dat*, *gbs*)
when *eq*(*vid*, *vid*(*op*)) = *true*

[la3] *tijd-laatst-uitgevoerde-opdracht*(*vid*, *gb*(*uitgevoerd*(*op*, *dat*)) ∼ *gbs*) =
tijd-laatst-uitgevoerde-opdracht(*vid*, *gbs*)
when *eq*(*vid*, *vid*(*op*)) = *false*

[la4] *tijd-laatst-uitgevoerde-opdracht*(*vid*, *gb*(*br*) ∼ *gbs*) = *tijd-laatst-uitgevoerde-opdracht*(*vid*, *gbs*)
when *eq*(*soort-bericht*(*br*), *nat*(^3)) = *false*

[la5] *tijd-laatst-uitgevoerde-opdracht-1*(*0vlis*, *dat*, *gbs*) = *startdatum*

[la6] *tijd-laatst-uitgevoerde-opdracht-1*(*vli* ∼ *vlis*, *dat*, *gbs*) =
max(*dat*, *tijd-laatst-uitgevoerde-opdracht*(*vid*(*vli*), *gbs*))

[po1] *preventieve-opdrachten*(*dat1*, *dat2*, *0gbs*) = *0opn*

[po2] *preventieve-opdrachten*(*dat1*, *dat2*, *gb*(*start*(*vli*)) ∼ *gbs*) =
preventieve-opdrachten(*max*(*dat1*, *tijd*(*vli*)), *dat2*, *vli*) ∼
preventieve-opdrachten(*dat1*, *dat2*, *gbs*)
when *tijd*(*vli*) < *dat2* = *true*

[po3] *preventieve-opdrachten*(*dat1*, *dat2*, *gb*(*stop*(*vid*, *dat3*)) ∼ *gbs*) =
preventieve-opdrachten-1(*dat1*, *min*(*dat2*, *dat3*), *vli*(*vid*, *gbs*)) ∼
preventieve-opdrachten(*dat1*, *dat2*, *verwijder-eerst-volgende-start*(*vid*, *gbs*))
when *dat1* < *dat3* = *true*

[po4] *preventieve-opdrachten*(*dat1*, *dat2*, *gb*(*start*(*vli*)) ∼ *gbs*) =
preventieve-opdrachten(*dat1*, *dat2*, *gbs*)
when *tijd*(*vli*) < *dat2* = *false*

[po5] *preventieve-opdrachten*(*dat1*, *dat2*, *gb*(*stop*(*vid*, *dat3*)) ∼ *gbs*) =
preventieve-opdrachten(*dat1*, *dat2*, *verwijder-eerst-volgende-start*(*vid*, *gbs*))
when *dat1* < *dat3* = *false*

[po6] *preventieve-opdrachten*(*dat1*, *dat2*, *gb*(*br*) ∼ *gbs*) = *preventieve-opdrachten*(*dat1*, *dat2*, *gbs*)
when *or*(*eq*(*soort-bericht*(*br*), *nat*(^1)), *eq*(*soort-bericht*(*br*), *nat*(^4))) = *false*

[po7] *preventieve-opdrachten-1*(*dat1*, *dat2*, *0vlis*) = *0opn*

[po8] *preventieve-opdrachten-1*(*dat1*, *dat2*, *vli* ∼ *vlis*) =
preventieve-opdrachten(*max*(*dat1*, *tijd*(*vli*)), *dat2*, *vli*)

[po9] *preventieve-opdrachten*(*dat1*, *dat2*, *vli*) = *0opn*
when *eq*(*sid*(*vli*), *sid*(^'I'^'H'^'G')) = *false*

[po10] *preventieve-opdrachten*(*dat1*, *dat2*, *vli*(*vid*, *configuratie*(*gn*, *k1*, *sid*(^'I'^'H'^'G')), *dat3*))
=
preventieve-opdrachten-IHG(*dat1*, *dat2*, *vid*, *dat3*,
schema(*vli*(*vid*, *configuratie*(*gn*, *k1*, *sid*(^'I'^'H'^'G')),
dat3)))

[po11] *preventieve-opdrachten-IHG*(*dat1*, *dat2*, *vid*, *dat3*, *schema-OBR*(*gdrTable*)) = *0opn*

[po12] *preventieve-opdrachten-IHG*(*dat1*, *dat2*, *vid*, *dat3*,
schema-IHG(*basiscyclus*, *leeg-GroepIdKjTable*))
= *0opn*

[po13] *preventieve-opdrachten-IHG*(*dat1*, *dat2*, *vid*, *dat3*,
schema-IHG(*basiscyclus*, *entry*(*gid*, *kgid*) ∼ *gkjTable*))
=
preventieve-opdrachten-IHG-1(*dat1*, *dat2*, *vid*, *dat3*, *basiscyclus*, *gid*, *kgid*, *nat*(^0)) ∼
preventieve-opdrachten-IHG(*dat1*, *dat2*, *vid*, *dat3*, *schema-IHG*(*basiscyclus*, *gkjTable*))

[po14] *preventieve-opdrachten-IHG-1*(*dat1*, *dat2*, *vid*, *dat3*, *basiscyclus*, *gid*, *kgid*, *tel*) = *0opn*
when *dat3* + (*tel* ∗ (*kgid* ∗ *basiscyclus*)) ≥ *dat2* = *true*

[po15] *preventieve-opdrachten-IHG-1*(*dat1*, *dat2*, *vid*, *dat3*, *basiscyclus*, *gid*, *kgid*, *tel*) =
preventieve-opdrachten-IHG-1(*dat1*, *dat2*, *vid*, *dat3*, *basiscyclus*, *gid*, *kgid*, *tel* + *nat*(^1))
when *dat3* + (*tel* ∗ (*kgid* ∗ *basiscyclus*)) < *dat1* = *true*

[po16] *preventieve-opdrachten-IHG-1*(*dat1*, *dat2*, *vid*, *dat3*, *basiscyclus*, *gid*, *kgid*, *tel*) =
preventief(*vid*, *gid*, *dat3* + (*tel* ∗ (*kgid* ∗ *basiscyclus*))) ∼
preventieve-opdrachten-IHG-1(*dat1*, *dat2*, *vid*, *dat3*, *basiscyclus*, *gid*, *kgid*,
tel + *nat*(ˆ1))
when *and*(*dat1* ≤ (*dat3* + (*tel* ∗ (*kgid* ∗ *basiscyclus*))),
dat3 + (*tel* ∗ (*kgid* ∗ *basiscyclus*)) < *dat2*)
= *true*

[r1] *preventief-alle-groepen*(*vli*) =
preventief-alle-groepen-1(*vid*(*vli*), *gids*(*groepen*(*configuratie*(*vli*))), *tijd*(*vli*))

[r2] *preventief-alle-groepen-1*(*vid*, *0gids*, *dat*) = *0opn*

[r3] *preventief-alle-groepen-1*(*vid*, *gid* ∼ *gids*, *dat*) =
preventief(*vid*, *gid*, *dat*) ∼ *preventief-alle-groepen-1*(*vid*, *gids*, *dat*)

[ba1] *bepaal-actie*(*vid*, *lids*, *dat*, *gbs*) = *bepaal-actie-0*(*vli*(*vid*, *gbs*), *lids*, *dat*, *gbs*)

[ba2] *bepaal-actie-0*(*0vlis*, *lids*, *dat*, *gbs*) = *0opn*

[ba3] *bepaal-actie-0*(*vli* ∼ *vlis*, *lids*, *dat*, *gbs*) = *bepaal-actie-1*(*vli*, *lids*, *dat*, *gbs*)

[ba4] *bepaal-actie-1*(*vli*, *lids*, *dat*, *gbs*) = *0opn*
when *or*(*eq*(*sid*(*vli*), *sid*(ˆ *'I'* ˆ *'H'* ˆ *'G'*)), *eq*(*sid*(*vli*), *sid*(ˆ *'O'* ˆ *'B'* ˆ *'R'*))) = *false*

[ba5] *bepaal-actie-1*(*vli*, *lids*, *dat*, *gbs*) = *bepaal-actie-2*(*vid*(*vli*), *isect*(*lids*(*vli*), *lids*), *dat*, *0lids*)
when *eq*(*sid*(*vli*), *sid*(ˆ *'I'* ˆ *'H'* ˆ *'G'*)) = *true*

[ba6] *bepaal-actie-1*(*vli*, *lids*, *dat*, *gbs*) =
bepaal-actie-OBR(*vid*(*vli*), *tabel*(*schema*(*vli*)), *isect*(*lids*(*vli*), *lids*), *dat*,
groepen(*configuratie*(*vli*)), *gbs*)
when *eq*(*sid*(*vli*), *sid*(ˆ *'O'* ˆ *'B'* ˆ *'R'*)) = *true*

[ba7] *bepaal-actie-OBR*(*vid*, *gdrTable*, *lids*, *dat*, *0gpn*, *gbs*) = *bepaal-actie-2*(*vid*, *lids*, *dat*, *0lids*)

[ba8] *bepaal-actie-OBR*(*vid*, *gdrTable*, *lids*, *dat*, *g1* ∼ *gn*, *gbs*) =
bepaal-actie-OBR-1(*vid*, *gdrTable*, *lids*, *dat*, *g1* ∼ *gn*, *lookup*(*gdrTable*, *gid*(*g1*)), *gbs*)

[ba9] *bepaal-actie-OBR-1*(*vid*, *gdrTable*, *lids*, *dat*, *g1* ∼ *gn*, *not-found*, *gbs*) = *0opn*

[ba10] *bepaal-actie-OBR-1*(*vid*, *gdrTable*, *lids*, *dat*, *g1* ∼ *gn*, *found*(*dr*), *gbs*) =
preventief(*vid*, *gid*(*g1*), *dat*) ∼
bepaal-actie-OBR-2(*vid*, *gdrTable*, *lids*, *dat*, *gn*, *lids*(*lpn*(*g1* ∼ *0gpn*)), *gbs*)
when *tijd-laatst-uitgevoerde-preventieve-opdracht*(*vid*, *gid*(*g1*), *gbs*) + *dr* ≤ *dat* = *true*

[ba11] *bepaal-actie-OBR-1*(*vid*, *gdrTable*, *lids*, *dat*, *g1* ∼ *gn*, *found*(*dr*), *gbs*) =
bepaal-actie-OBR(*vid*, *gdrTable*, *lids*, *dat*, *gn*, *gbs*)
when *tijd-laatst-uitgevoerde-preventieve-opdracht*(*vid*, *gid*(*g1*), *gbs*) + *dr* ≤ *dat* = *false*

[ba12] *bepaal-actie-OBR-2*(*vid*, *gdrTable*, *lids*, *dat*, *0gpn*, *lids'*, *gbs*) =
bepaal-actie-2(*vid*, *lids*, *dat*, *lids'*)

[ba13] *bepaal-actie-OBR-2*(*vid*, *gdrTable*, *lids*, *dat*, *g1* ∼ *gn*, *lids'*, *gbs*) =
bepaal-actie-OBR-3(*vid*, *gdrTable*, *lids*, *dat*, *g1* ∼ *gn*, *lookup*(*gdrTable*, *gid*(*g1*)), *lids'*, *gbs*)

[ba14] *bepaal-actie-OBR-3*(*vid*, *gdrTable*, *lids*, *dat*, *g1* ∼ *gn*, *not-found*, *lids'*, *gbs*) = *0opn*

[ba15] *bepaal-actie-OBR-3*(*vid*, *gdrTable*, *lids*, *dat*, *g1* ∼ *gn*, *found*(*dr*), *lids'*, *gbs*) =
preventief(*vid*, *gid*(*g1*), *dat*) ∼
bepaal-actie-OBR-2(*vid*, *gdrTable*, *lids*, *dat*, *gn*, *lids*(*lpn*(*g1* ∼ *0gpn*)) ∼ *lids'*, *gbs*)
when *tijd-laatst-uitgevoerde-preventieve-opdracht*(*vid*, *gid*(*g1*), *gbs*) + *dr* ≤ *dat* = *true*

[ba16] *bepaal-actie-OBR-3*(*vid*, *gdrTable*, *lids*, *dat*, *g1* ∼ *gn*, *found*(*dr*), *lids'*, *gbs*) =
bepaal-actie-OBR-2(*vid*, *gdrTable*, *lids*, *dat*, *gn*, *lids'*, *gbs*)
when *tijd-laatst-uitgevoerde-preventieve-opdracht*(*vid*, *gid*(*g1*), *gbs*) + *dr* ≤ *dat* = *false*

[ba17] *bepaal-actie-2*(*vid*, *0lids*, *dat*, *lids'*) = *0opn*

[ba18] *bepaal-actie-2*(*vid*, *lid* ∼ *lids*, *dat*, *lids'*) =
correctief(*vid*, *lid*, *dat*) ∼ *bepaal-actie-2*(*vid*, *lids*, *dat*, *lids'*)
when *element*(*lid*, *lids'*) = *false*

[ba19] *bepaal-actie-2*(*vid*, *lid* ∼ *lids*, *dat*, *lids'*) = *bepaal-actie-2*(*vid*, *lids*, *dat*, *lids'*)
when *element*(*lid*, *lids'*) = *true*

[m1] *merge-opdrachten*(*opn*, *opn'*) = *opn* ∼ *opn'*

end *LRdata*

15.4 Modulen voor simulatie en procesdefinitie

In deze paragraaf staan modulen met een ondersteunend karakter. Ten eerste, `Parameters`, is op te vatten als een instellingenbestand. (Het zou beter zijn om deze module als parameter te binden in plaats van de huidige import.) De daarop volgende modulen `Simulatie-vergelijkingen` en `Simulatie-verzamelingen` dienen zuiver een alleen voor simulatie doeleinden. Deze twee modulen zijn integraal opgenomen om een idee te geven hoe dergelijke modulen t.b.v. simulatie eruit kunnen zien. De laatste twee modulen `Procesnaam` en `Acties-en-Communicatie` dienen voor de definitie van de acties en communicatie tussen de processen in Paragraaf 15.5.

data module *Parameters*
begin
exports
begin
functions
prijs : *Lamptype* → *Kosten*
vervangtarief : *LichtpuntLocatie* → *Kosten*
verdeling : *Lamptype* → *LevensduurVerdeling*
startdatum : → *Datum*
end
imports *Tijd*[61], *LichtpuntLocatie*[60], *Lamptype*[62], *Kosten*[67], *LevensduurVerdeling*[62]
equations
[p1] $prijs(Halogeen10V) = k(\hat{}1 \hat{} 4 \hat{} 7 \hat{} 4)$
[p2] $prijs(230V75W) = k(\hat{}2 \hat{} 5 \hat{} 3)$
[p3] $prijs(40V40W) = k(\hat{}8 \hat{} 5 \hat{} 3)$
[v1] $vervangtarief(laag) = k(\hat{}4 \hat{} 0 \hat{} 0)$
[v2] $vervangtarief(hoog) = k(\hat{}1 \hat{} 9 \hat{} 0 \hat{} 0)$
[d1] $verdeling(Halogeen10V) = weibull(vw(\hat{}3, \hat{}1 \hat{} 5 \hat{} 1), sw(\hat{}0, \hat{}0 \hat{} 0 \hat{} 0 \hat{} 1 \hat{} 0 \hat{} 3 \hat{} 5))$
[d2] $verdeling(230V75W) = weibull(vw(\hat{}3, \hat{}6 \hat{} 0 \hat{} 5), sw(\hat{}0, \hat{}0 \hat{} 0 \hat{} 0 \hat{} 1 \hat{} 1 \hat{} 2 \hat{} 9))$
[d3] $verdeling(40V40W) = weibull(vw(\hat{}5, \hat{}9 \hat{} 1 \hat{} 4), sw(\hat{}0, \hat{}0 \hat{} 0 \hat{} 0 \hat{} 1 \hat{} 1 \hat{} 7 \hat{} 5))$
[sd] $startdatum = datum(nat(\hat{}0))$
end *Parameters*

data module *Simulatie-vergelijkingen*
begin
exports
begin
functions
cfa : → *Configuratie*
cfb : → *Configuratie*
cfc : → *Configuratie*
vlia1 : → *Vli*
vlib0 : → *Vli*
vlib2 : → *Vli*
vlic : → *Vli*
end
imports *LRdata*[80], *Betrouwbaarheid*[72], *Exploitatie*[73]
variables *cf* : → *Configuratie*
equations

[ca] *cfa* =
configuratie(*groep*(*gid*(^ 'a'),
lichtpunt(*lid*(^ 'a'), *40V40W*, *laag*, *On*) ∼
(*lichtpunt*(*lid*(^ 'b'), *40V40W*, *laag*, *On*) ∼ *0lpn*),
k(^1), *k*(^1), *k*(^1), *On*)
∼
(*groep*(*gid*(^ 'b'), *lichtpunt*(*lid*(^ 'c'), *230V75W*, *hoog*, *bk*(^5)) ∼ *0lpn*, *k*(^1),
k(^1), *k*(^1), *bk*(^5))
∼ *0gpn*),
k(^1), *sid*(^ 'I' ^ 'H' ^ 'G'))

[cb] *cfb* =
configuratie(*groep*(*gid*(^ 'c'),
lichtpunt(*lid*(^ 'd'), *40V40W*, *hoog*, *bk*(^8)) ∼
(*lichtpunt*(*lid*(^ 'e'), *40V40W*, *hoog*, *bk*(^9)) ∼ *0lpn*),
k(^1), *k*(^1), *k*(^1), *bk*(^9))
∼
(*groep*(*gid*(^ 'd'), *lichtpunt*(*lid*(^ 'f'), *230V75W*, *laag*, *bk*(^5)) ∼ *0lpn*, *k*(^1),
k(^1), *k*(^1), *bk*(^5))
∼ *0gpn*),
k(^1), *sid*(^ 'O' ^ 'B' ^ 'R'))

[cc] *cfc* =
configuratie(*groep*(*gid*(^ 'c'), *0lpn*, *k*(^1), *k*(^1), *k*(^1), *bk*(^9)) ∼ *0gpn*, *k*(^1),
sid(^ 'O' ^ 'B' ^ 'R'))

[va1] *vlia1* = *vli*(*vid*(^ 'a'), *cfa*, *datum*(*nat*(^1)))

[vb0] *vlib0* = *vli*(*vid*(^ 'b'), *cfb*, *startdatum*)

[vb2] *vlib2* = *vli*(*vid*(^ 'b'), *cfb*, *datum*(*nat*(^2)))

[vc] *vlic* = *vli*(*vid*(^ 'b'), *cfc*, *startdatum*)

[s1] *schema*(*cf*) =
schema-IHG(*duur1*(*nat*(^0)),
entry(*gid*(^ 'a'), *kj*(*nat*(^1))) ∼
(*entry*(*gid*(^ 'b'), *kj*(*nat*(^3))) ∼ *leeg-GroepIdKjTable*))
when *cf* = *cfa*

[s2] *schema*(*cf*) =
schema-OBR(*entry*(*gid*(^ 'c'), *duur1*(*nat*(^3))) ∼
(*entry*(*gid*(^ 'd'), *duur1*(*nat*(^2))) ∼ *leeg-GroepIdDuurTable*))
when *cf* = *cfb*

[e1] *exploitatie*(*cf*) = *exploitatie*(*k*(^1), *duur1*(*nat*(^0))) **when** *cf* = *cfa*

[e2] *exploitatie*(*cf*) = *exploitatie*(*k*(^1), *duur1*(*nat*(^0))) **when** *cf* = *cfb*

[b1] *betrouwbaarheid*(*cf*) = *betrouwbaarheid*(*make-fp*(^0, ^1), *duur1*(*nat*(^0))) **when** *cf* = *cfa*

[b2] *betrouwbaarheid*(*cf*) = *betrouwbaarheid*(*make-fp*(^0, ^1), *duur1*(*nat*(^0))) **when** *cf* = *cfb*

end *Simulatie-vergelijkingen*

process module *Simulatie-verzamelingen*
begin
exports
begin
sets
of *Duur1*
Duur1-f = {*duur1*(*nat*(^0)), *duur1*(*nat*(^1)), *duur1*(*nat*(^2)), *duur1*(*nat*(^3)),
duur1(*nat*(^4)), *duur1*(*nat*(^ 5))}
of *Datum*

Datum-f = {*datum*(*nat*(^0)), *datum*(*nat*(^1)), *datum*(*nat*(^2)), *datum*(*nat*(^3)),
datum(*nat*(^4)), *datum*(*nat*(^5)), *datum*(*nat*(^6)), *datum*(*nat*(^7)),
datum(*nat*(^8)), *datum*(*nat*(^9)), *datum*(*nat*(^1 ^ 0)), *datum*(*nat*(^1 ^ 1)),
datum(*nat*(^1 ^ 2)), *datum*(*nat*(^1 ^ 3)), *datum*(*nat*(^1 ^ 4))}

of *VliId*

VliId-f = {*vid*(^ 'a'), *vid*(^ 'b'), *vid*(^ 'c')}

of *LichtpuntIds*

LichtpuntIds-f = {*lid*(^ 'a') ∼ *0lids*, *lid*(^ 'b') ∼ *0lids*, *lid*(^ 'c') ∼ *0lids*, *lid*(^ 'd') ∼ *0lids*,
lid(^ 'e') ∼ *0lids*, *lid*(^ 'f') ∼ *0lids*, *lid*(^ 'a') ∼ (*lid*(^ 'b') ∼ *0lids*),
lid(^ 'd') ∼ (*lid*(^ 'e') ∼ *0lids*)}

of *Opdracht*

Opdracht-f = {*correctief*(*vid*(^ 'a'), *lid*(^ 'a'), *datum*(*nat*(^2))),
preventief(*vid*(^ 'a'), *gid*(^ 'a'), *datum*(*nat*(^1))),
preventief(*vid*(^ 'a'), *gid*(^ 'a'), *datum*(*nat*(^3))),
preventief(*vid*(^ 'a'), *gid*(^ 'a'), *datum*(*nat*(^5))),
preventief(*vid*(^ 'a'), *gid*(^ 'a'), *datum*(*nat*(^7))),
preventief(*vid*(^ 'a'), *gid*(^ 'a'), *datum*(*nat*(^9))),
preventief(*vid*(^ 'a'), *gid*(^ 'a'), *datum*(*nat*(^1 ^ 1))),
preventief(*vid*(^ 'a'), *gid*(^ 'a'), *datum*(*nat*(^1 ^ 3))),
preventief(*vid*(^ 'a'), *gid*(^ 'b'), *datum*(*nat*(^1))),
preventief(*vid*(^ 'a'), *gid*(^ 'b'), *datum*(*nat*(^5))),
preventief(*vid*(^ 'a'), *gid*(^ 'b'), *datum*(*nat*(^9))),
preventief(*vid*(^ 'a'), *gid*(^ 'b'), *datum*(*nat*(^1 ^ 3))),
preventief(*vid*(^ 'b'), *gid*(^ 'c'), *datum*(*nat*(^2))),
preventief(*vid*(^ 'b'), *gid*(^ 'd'), *datum*(*nat*(^2))),
preventief(*vid*(^ 'b'), *gid*(^ 'c'), *datum*(*nat*(^4))),
correctief(*vid*(^ 'b'), *lid*(^ 'f'), *datum*(*nat*(^4)))}

of *Configuratie*

Configuratie-f = {*cfa*, *cfb*, *cfc*}

of *Vli*

Vli-f = {*vlia1*, *vlib0*, *vlib2*, *vlic*}

of *Schema*

Schema-f = {*schema*(*cfa*), *schema*(*cfb*)}

of *Betrouwbaarheid*

Betrouwbaarheid-f = {*betrouwbaarheid*(*cfa*), *betrouwbaarheid*(*cfb*)}

of *Exploitatie*

Exploitatie-f = {*exploitatie*(*cfa*), *exploitatie*(*cfb*)}

end

imports *Procesnaam*[86], *Simulatie-vergelijkingen*[84]

end *Simulatie-verzamelingen*

data module *Procesnaam*

begin

exports

begin

sorts *Procesnaam*

functions

OP : → *Procesnaam*

CA : → *Procesnaam*

AN : → *Procesnaam*

ED : → *Procesnaam*

SB : → *Procesnaam*
IF : → *Procesnaam*
RP : → *Procesnaam*
SA : → *Procesnaam*
DB : → *Procesnaam*
BO : → *Procesnaam*
KL : → *Procesnaam*

end
end *Procesnaam*

process module *Acties-en-Communicatie*
begin
exports
begin
atoms
foutmelding : *Bericht*
toon : *Schema # Betrouwbaarheid # Exploitatie*
invoer : *Datum*
invoer : *Bericht*
read, send, c : *Procesnaam # Procesnaam # Configuratie*
read, send, c : *Procesnaam # Procesnaam # Opdrachten*
read, send, c : *Procesnaam # Procesnaam # Gebeurtenissen*
read, send, c : *Procesnaam # Procesnaam # Schema*
read, send, c : *Procesnaam # Procesnaam # Vlis*
read, send, c : *Procesnaam # Procesnaam # Vli*
read, send, c : *Procesnaam # Procesnaam # Datum*
read, send, c : *Procesnaam # Procesnaam # VliId*
read, send, c : *Procesnaam # Procesnaam # VliIds*
read, send, c : *Procesnaam # Procesnaam # VliId # Configuratie*
read, send, c : *Procesnaam # Procesnaam # Bericht*
read, send, c : *Procesnaam # Procesnaam # BOOLEAN*
read, send, c : *Procesnaam # Procesnaam # Schema # Betrouwbaarheid # Exploitatie*
sets
of atoms

$H = \{read(pn, pn', cf), send(pn', pn, cf)$
$\mid pn$ **in** $Procesnaam, pn'$ **in** $Procesnaam, cf$ **in** $Configuratie\} +$
$\{read(pn, pn', opn), send(pn', pn, opn)$
$\mid pn$ **in** $Procesnaam, pn'$ **in** $Procesnaam, opn$ **in** $Opdrachten\} +$
$\{read(pn, pn', gbs), send(pn', pn, gbs)$
$\mid pn$ **in** $Procesnaam, pn'$ **in** $Procesnaam, gbs$ **in** $Gebeurtenissen\} +$
$\{read(pn, pn', sch), send(pn', pn, sch)$
$\mid pn$ **in** $Procesnaam, pn'$ **in** $Procesnaam, sch$ **in** $Schema\} +$
$\{read(pn, pn', vlis), send(pn', pn, vlis)$
$\mid pn$ **in** $Procesnaam, pn'$ **in** $Procesnaam, vlis$ **in** $Vlis\} +$
$\{read(pn, pn', vli), send(pn', pn, vli)$
$\mid pn$ **in** $Procesnaam, pn'$ **in** $Procesnaam, vli$ **in** $Vli\} +$
$\{read(pn, pn', dat), send(pn', pn, dat)$
$\mid pn$ **in** $Procesnaam, pn'$ **in** $Procesnaam, dat$ **in** $Datum\} +$
$\{read(pn, pn', vid), send(pn', pn, vid)$
$\mid pn$ **in** $Procesnaam, pn'$ **in** $Procesnaam, vid$ **in** $VliId\} +$
$\{read(pn, pn', vids), send(pn', pn, vids)$
$\mid pn$ **in** $Procesnaam, pn'$ **in** $Procesnaam, vids$ **in** $VliIds\} +$
$\{read(pn, pn', vid, cf), send(pn', pn, vid, cf)$
$\mid pn$ **in** $Procesnaam, pn'$ **in** $Procesnaam, vid$ **in** $VliId, cf$ **in** $Configuratie\} +$
$\{read(pn, pn', br), send(pn', pn, br)$
$\mid pn$ **in** $Procesnaam, pn'$ **in** $Procesnaam, br$ **in** $Bericht\} +$
$\{read(pn, pn', bool), send(pn', pn, bool)$
$\mid pn$ **in** $Procesnaam, pn'$ **in** $Procesnaam, bool$ **in** $BOOLEAN\} +$
$\{read(pn, pn', sch, rel, expl), send(pn', pn, sch, rel, expl)$
$\mid pn$ **in** $Procesnaam, pn'$ **in** $Procesnaam, sch$ **in** $Schema, rel$ **in** $Betrouwbaarheid,$
$expl$ **in** $Exploitatie\}$

end

imports *Simulatie-verzamelingen*[85]

communications

$read(pn, pn', cf) \mid send(pn', pn, cf) = c(pn, pn', cf)$
for pn **in** $Procesnaam, pn'$ **in** $Procesnaam, cf$ **in** $Configuratie$
$read(pn, pn', opn) \mid send(pn', pn, opn) = c(pn, pn', opn)$
for pn **in** $Procesnaam, pn'$ **in** $Procesnaam, opn$ **in** $Opdrachten$
$read(pn, pn', gbs) \mid send(pn', pn, gbs) = c(pn, pn', gbs)$
for pn **in** $Procesnaam, pn'$ **in** $Procesnaam, gbs$ **in** $Gebeurtenissen$
$read(pn, pn', sch) \mid send(pn', pn, sch) = c(pn, pn', sch)$
for pn **in** $Procesnaam, pn'$ **in** $Procesnaam, sch$ **in** $Schema$
$read(pn, pn', vlis) \mid send(pn', pn, vlis) = c(pn, pn', vlis)$
for pn **in** $Procesnaam, pn'$ **in** $Procesnaam, vlis$ **in** $Vlis$
$read(pn, pn', vli) \mid send(pn', pn, vli) = c(pn, pn', vli)$
for pn **in** $Procesnaam, pn'$ **in** $Procesnaam, vli$ **in** Vli
$read(pn, pn', dat) \mid send(pn', pn, dat) = c(pn, pn', dat)$
for pn **in** $Procesnaam, pn'$ **in** $Procesnaam, dat$ **in** $Datum$
$read(pn, pn', vid) \mid send(pn', pn, vid) = c(pn, pn', vid)$
for pn **in** $Procesnaam, pn'$ **in** $Procesnaam, vid$ **in** $VliId$
$read(pn, pn', vids) \mid send(pn', pn, vids) = c(pn, pn', vids)$
for pn **in** $Procesnaam, pn'$ **in** $Procesnaam, vids$ **in** $VliIds$
$read(pn, pn', vid, cf) \mid send(pn', pn, vid, cf) = c(pn, pn', vid, cf)$
for pn **in** $Procesnaam, pn'$ **in** $Procesnaam, vid$ **in** $VliId, cf$ **in** $Configuratie$
$read(pn, pn', br) \mid send(pn', pn, br) = c(pn, pn', br)$
for pn **in** $Procesnaam, pn'$ **in** $Procesnaam, br$ **in** $Bericht$
$read(pn, pn', bool) \mid send(pn', pn, bool) = c(pn, pn', bool)$
for pn **in** $Procesnaam, pn'$ **in** $Procesnaam, bool$ **in** $BOOLEAN$

read(*pn*, *pn'*, *sch*, *rel*, *expl*) | *send*(*pn'*, *pn*, *sch*, *rel*, *expl*) = *c*(*pn*, *pn'*, *sch*, *rel*, *expl*)
for *pn* **in** *Procesnaam*, *pn'* **in** *Procesnaam*, *sch* **in** *Schema*, *rel* **in** *Betrouwbaarheid*,
expl **in** *Exploitatie*
end *Acties-en-Communicatie*

15.5 Processen

Hieronder volgen negen procesmodulen. De rol van de eerste acht is kort besproken in Paragraaf 11. De laatste module is de topmodule van de gehele specificatie, hier wordt het proces `BOS` gespecificeerd.

process module *Klok*
begin
exports
begin
atoms *tik*
processes *Klok* : *Datum*
end
imports *Acties-en-Communicatie*[87]
processes *Klok'* : *Datum*
variables *nu* : → *Datum*
definitions
Klok(*nu*) = *send*(*KL*, *RP*, *tijd*(*nu*)) · *Klok'*(*nu*)
Klok'(*nu*) =
(*send*(*KL*, *RP*, *tijd*(*nu*)) + *send*(*KL*, *BO*, *tijd*(*nu*)) + *send*(*KL*, *SB*, *tijd*(*nu*))) · *Klok'*(*nu*)
+ *tik* · *Klok*(*nu* + *duur1*(*nat*(ˆ0)))
end *Klok*

process module *Interface*
begin
exports
begin
atoms
invoer : *VliId*
toon-opdrachten : *Opdrachten*
processes *Interface*
end
imports *Acties-en-Communicatie*[87]
definitions
Interface =

```
      (   sum(vid in VliId-f, invoer(vid) · send(IF, SA, vid))
       + sum(vid in VliId-f,
              sum(lids in LichtpuntIds-f,
                   sum(dat in Datum-f,
                        invoer(defect(vid, lids, dat)) · send(IF, BO, defect(vid, lids, dat)))))
       + sum(op in Opdracht-f,
              sum(dat in Datum-f, invoer(uitgevoerd(op, dat)) · send(IF, RP, uitgevoerd(op, dat))))
       + sum(vid in VliId-f,
              invoer(analyse-configuratie(vid)) · send(IF, CA, analyse-configuratie(vid)))
       + invoer(analyse-nieuwe-configuratie) · send(IF, CA, analyse-nieuwe-configuratie)
       + sum(opn in Opdrachten, read(IF, RP, opn) · toon-opdrachten(opn))
       + sum(vid in VliId-f,
              sum(dat in Datum-f, invoer(stop(vid, dat)) · send(IF, SB, stop(vid, dat))))) ·
      Interface
end Interface

process module Database
begin
  exports
  begin
    processes Database : Opdrachten # Gebeurtenissen
  end
  imports Acties-en-Communicatie[87]
  variables
    opn : → Opdrachten
    gbs : → Gebeurtenissen
  definitions
  Database(opn, gbs) =
       sum(op in Opdracht-f,
            sum(dat in Datum-f,
                 read(DB, RP, uitgevoerd(op, dat)) ·
                 Database(verwijder(op, opn), gb(uitgevoerd(op, dat)) ~ gbs)))
    + sum(vli in Vli-f, read(DB, RP, start(vli)) · Database(opn, gb(start(vli)) ~ gbs))
    + sum(vid in VliId-f,
            sum(dat in Datum-f,
                 read(DB, RP, stop(vid, dat)) · Database(opn, gb(stop(vid, dat)) ~ gbs)))
    + sum(vid in VliId-f,
            sum(lids in LichtpuntIds-f,
                 sum(dat in Datum-f,
                      read(DB, BO, defect(vid, lids, dat)) ·
                      Database(opn, gb(defect(vid, lids, dat)) ~ gbs))))
    + sum(dat1 in Datum-f,
            sum(dat2 in Datum-f,
                 read(DB, RP, bepaal-preventieve-opdrachten(dat1, dat2)) ·
                 Database(preventieve-opdrachten(dat1, dat2, gbs) ~ opn, gbs)))
    + sum(dat1 in Datum-f,
            sum(dat2 in Datum-f,
                 sum(vli in Vli-f,
                      read(DB, RP, bepaal-preventieve-opdrachten(dat1, dat2, vli)) ·
                      Database(preventieve-opdrachten(dat1, dat2, vli) ~ opn, gbs))))
```

```
    + sum(vli in Vli-f,
          read(DB, RP, preventieve-remplace-alle-groepen(vli)) ·
          Database(preventief-alle-groepen(vli) ~ opn, gbs))
    + sum(vid in VliId-f,
          sum(dat in Datum-f,
              read(DB, RP, verwijder-opdrachten(vid, dat)) ·
              Database(verwijder-opdrachten(vid, dat, opn), gbs)))
    + sum(vid in VliId-f,
          sum(lids in LichtpuntIds-f,
              sum(dat in Datum-f,
                  read(DB, BO, bepaal-actie(vid, lids, dat)) ·
                  Database(merge-opdrachten(bepaal-actie(vid, lids, dat, gbs), opn), gbs))))
    + (   sum(vid in VliId-f,
              read(DB, SB, tijd-laatst-uitgevoerde-opdracht(vid)) ·
              send(DB, SB, tijd-laatst-uitgevoerde-opdracht(vid, gbs)))
      + sum(vid in VliId-f, read(DB, BO, bekend(vid)) · send(DB, BO, bekend(vid, gbs)))
      + sum(vid in VliId-f,
            sum(lids in LichtpuntIds-f,
                read(DB, BO, onderdeel-van(lids, vid)) ·
                (   [onderdeel-van(lids, vid, gbs) = true] →
                    send(DB, BO, onderdeel-van(lids, vid))
                 + [onderdeel-van(lids, vid, gbs) = false] →
                    send(DB, BO, geen-onderdeelvan(lids, vid)))))
      + sum(vid in VliId-f,
            sum(dat in Datum-f,
                read(DB, BO, onder-schema(vid, dat)) ·
                send(DB, BO, onder-schema(vid, dat, gbs))))
      + sum(vid in VliId-f, read(DB, BO, geef-vli(vid)) · send(DB, BO, vli(vid, gbs)))
      + sum(vid in VliId-f, read(DB, CA, geef-vli(vid)) · send(DB, CA, vli(vid, gbs)))
      + sum(vid in VliId-f,
            read(DB, SB, start-of-stop(vid)) · send(DB, SB, start-of-stop(vid, gbs)))
      + sum(vid in VliId-f,
            read(DB, BO, start-of-stop(vid)) · send(DB, BO, start-of-stop(vid, gbs)))
      + send(DB, BO, gbs)
      + send(DB, RP, gbs)
      + send(DB, RP, opn)) ·
    Database(opn, gbs)
end Database

process module Configuratie-Analyse
begin
  exports
  begin
    atoms
      edit              : VliId # Configuratie
      toevoegen
      niet-toevoegen
    processes
      Optimalisator
      ConfiguratieAnalyse
  end
  imports Acties-en-Communicatie[87]
```

processes

Editor

Editor : *VliId* # *Configuratie*

Analyse

Analyse1 : *VliId* # *Configuratie*

variables

vid : → *VliId*

cf : → *Configuratie*

definitions

ConfiguratieAnalyse =

read(*CA*, *IF*, *analyse-nieuwe-configuratie*) · (*Analyse* || *Editor*) · *ConfiguratieAnalyse*

+ **sum**(*vid* **in** *VliId-f*,

read(*CA*, *IF*, *analyse-configuratie*(*vid*)) · *send*(*CA*, *DB*, *geef-vli*(*vid*)) ·

(*read*(*CA*, *DB*, *0vlis*) · *foutmelding*(*vli-bestaat-niet*)

+ **sum**(*vli* **in** *Vli-f*,

sum(*vlis* **in** *Vlis*,

read(*CA*, *DB*, *vli* ∼ *vlis*) · (*Analyse* || *Editor*(*vid*, *configuratie*(*vli*)))))))) ·

ConfiguratieAnalyse

Editor =

sum(*vid* **in** *VliId-f*,

sum(*cf* **in** *Configuratie-f*,

edit(*vid*, *cf*) · *send*(*ED*, *AN*, *vid*, *cf*) ·

(*read*(*ED*, *AN*, *stop*) + *read*(*ED*, *AN*, *ga-door*) · *Editor*(*vid*, *cf*))))

+ *send*(*ED*, *AN*, *stop*)

Editor(*vid*, *cf*) =

sum(*vid* **in** *VliId-f*,

sum(*cf'* **in** *Configuratie-f*,

edit(*vid*, *cf'*) · *send*(*ED*, *AN*, *vid*, *cf'*) ·

(*read*(*ED*, *AN*, *stop*) + *read*(*ED*, *AN*, *ga-door*) · *Editor*(*vid*, *cf'*))))

+ *send*(*ED*, *AN*, *stop*)

Analyse =

sum(*vid* **in** *VliId-f*,

sum(*cf* **in** *Configuratie-f*,

read(*AN*, *ED*, *vid*, *cf*) ·

([*vorm-eis*(*cf*) = *true*] → *send*(*AN*, *OP*, *cf*) · *Analyse1*(*vid*, *cf*)

+ [*vorm-eis*(*cf*) = *false*] → *foutmelding*(*editor-inhoud-niet-consistent*) ·

send(*AN*, *ED*, *ga-door*) · *Analyse*)))

+ *read*(*AN*, *ED*, *stop*)

Analyse1(*vid*, *cf*) =

sum(*sch* **in** *Schema-f*,

sum(*rel* **in** *Betrouwbaarheid-f*,

sum(*expl* **in** *Exploitatie-f*, *read*(*AN*, *OP*, *sch*, *rel*, *expl*) · *toon*(*sch*, *rel*, *expl*)))) ·

(*toevoegen* · **sum**(*dat* **in** *Datum-f*, *invoer*(*dat*) · *send*(*AN*, *SB*, *start*(*vli*(*vid*, *cf*, *dat*)))) ·

(*read*(*AN*, *SB*, *oke*) · *send*(*AN*, *ED*, *stop*)

+ *read*(*AN*, *SB*, *niet-oke*) · *send*(*AN*, *ED*, *ga-door*) · *Analyse*)

+ *niet-toevoegen* · *send*(*AN*, *ED*, *stop*))

+ *read*(*AN*, *OP*, *geen-schema-gevonden*) · *foutmelding*(*geen-schema-gevonden*) ·

send(*AN*, *ED*, *ga-door*) · *Analyse*

Optimalisator =

```
    sum(cf in Configuratie-f,
          read(OP, AN, cf) ·
          (   send(OP, AN, schema(cf), betrouwbaarheid(cf), exploitatie(cf))
           + send(OP, AN, geen-schema-gevonden))) ·
    Optimalisator
end Configuratie-Analyse

process module SchemaBeheer
begin
  exports
  begin
    atoms toon : Opdrachten
    processes SchemaBeheer
  end
  imports Acties-en-Communicatie[87]
  processes
    SchemaBeheer1 : Datum # VliId # Configuratie # Datum
    SchemaBeheer2 : Datum # VliId
    SchemaBeheer3 : Datum # VliId # Configuratie # Datum
  variables
    dat, nu, starttijd : → Datum
    vid                : → VliId
    cf                 : → Configuratie
  definitions
  SchemaBeheer =
      sum(vid in VliId-f,
            sum(cf in Configuratie-f,
                  sum(dat in Datum-f,
                        read(SB, AN, start(vli(vid, cf, dat))) ·
                        sum(nu in Datum-f,
                              read(SB, KL, tijd(nu)) · send(SB, DB, start-of-stop(vid)) ·
                              SchemaBeheer1(nu, vid, cf, dat)))))
    + sum(vid in VliId-f,
            sum(dat in Datum-f,
                  read(SB, IF, stop(vid, dat)) · send(SB, DB, start-of-stop(vid)) ·
                  SchemaBeheer2(dat, vid)))
  SchemaBeheer1(nu, vid, cf, dat) =
    (   sum(dat' in Datum-f,
              read(SB, DB, stop(vid, dat')) ·
              (   [(dat > dat') = true] → SchemaBeheer3(nu, vid, cf, dat)
               + [(dat > dat') = false] →
                    (foutmelding(Starttijdstip-voor-Stoptijdstip) || send(SB, AN, niet-oke)) ·
                  SchemaBeheer))
     + read(SB, DB, triviaal-bericht) · SchemaBeheer3(nu, vid, cf, dat)
     + sum(vli in Vli-f,
              read(SB, DB, start(vli)) ·
              (foutmelding(Geen-stop-van-huidig-schema-opgegeven) || send(SB, AN, niet-oke)) ·
              SchemaBeheer))
  SchemaBeheer2(dat, vid) =
```

```
    (   sum(dat' in Datum-f, read(SB, DB, stop(vid, dat')) · foutmelding(vli-is-reeds-gestopt(vid)))
     + read(SB, DB, triviaal-bericht) · foutmelding(vli-bestaat-niet)
     + sum(vli in Vli-f,
             read(SB, DB, start(vli)) ·
             (   [(dat > tijd(vli)) = true] → send(SB, DB, tijd-laatst-uitgevoerde-opdracht(vid)) ·
                 sum(dat' in Datum-f,
                       read(SB, DB, dat') ·
                       (   [(dat > dat') = true] → send(SB, RP, stop(vid, dat))
                        + [(dat > dat') = false] → foutmelding(stoptijdstip-conflict-actie)))
              + [(dat > tijd(vli)) = false] → foutmelding(stoptijdstip-conflict-start)))) ·
    SchemaBeheer
  SchemaBeheer3(nu, vid, cf, starttijd) =
    (   [(starttijd > nu) = true] → (send(SB, RP, start(vli(vid, cf, starttijd))) || send(SB, AN, oke))
     + [(starttijd > nu) = false] →
         (foutmelding(starttijd-schema-in-het-verleden) || send(SB, AN, niet-oke))) ·
    SchemaBeheer
end SchemaBeheer

process module RemplacePlanning
begin
  exports
  begin
    processes RemplacePlanning : Datum # Duur1
  end
  imports Acties-en-Communicatie[87]
  processes
    RemplacePlanning1 : Datum # Duur1 # Opdracht # Datum # Datum
    RemplacePlanning2 : Datum # Duur1
  variables
    tot, dat, nu : → Datum
    horizon      : → Duur1
    op           : → Opdracht
  definitions
  RemplacePlanning(tot, horizon) =
      sum(nu in Datum-f,
            read(RP, KL, tijd(nu)) ·
            (   [and(nu < tot, tot < (nu + horizon + een)) = true] →
                  send(RP, DB, bepaal-preventieve-opdrachten(tot, (nu + horizon + een))) ·
                RemplacePlanning2((nu + horizon + een), horizon)
             + [(nu < tot) = false] → foutmelding(Planning-update-te-laat)
             + [(tot ≥ (nu + horizon + een)) = true] → skip · RemplacePlanning(tot, horizon)))
    + sum(op in Opdracht-f,
            sum(dat in Datum-f,
                  read(RP, IF, uitgevoerd(op, dat)) ·
                  sum(nu in Datum-f,
                        read(RP, KL, tijd(nu)) ·
                        sum(opn in Opdrachten,
                              read(RP, DB, opn) ·
                              (   [element(op, opn) = true] →
                                    RemplacePlanning1(tot, horizon, op, dat, nu)
                               + [element(op, opn) = false] → foutmelding(opdracht-onbekend(op)) ·
                                 RemplacePlanning(tot, horizon))))))
```

```
    + sum(vli in Vli-f,
            read(RP, SB, start(vli)) ·
            sum(nu in Datum-f,
                  read(RP, KL, tijd(nu)) ·
                  (   [(tijd(vli) ≥ nu) = true] → send(RP, DB, start(vli)) ·
                      (   [sid(vli) = sid(^ 'I' ^ 'H' ^ 'G')] →
                            send(RP, DB, bepaal-preventieve-opdrachten(tijd(vli), tot, vli))
                       + [sid(vli) = sid(^ 'O' ^ 'B' ^ 'R')] →
                            send(RP, DB, preventieve-remplace-alle-groepen(vli))) ·
                      RemplacePlanning2(tot, horizon)
                   + [(tijd(vli) ≥ nu) = false] → foutmelding(Opdracht-mogelijk-geantidateerd))))
    + sum(vid in VliId-f,
            sum(dat in Datum-f,
                  read(RP, SB, stop(vid, dat)) ·
                  (send(RP, DB, stop(vid, dat)) || send(RP, DB, verwijder-opdrachten(vid, dat)))) ·
      RemplacePlanning2(tot, horizon)
    + RemplacePlanning2(tot, horizon)

  RemplacePlanning1(tot, horizon, op, dat, nu) =
      [correctief(op) = true] →
        (   [and(dat > tijd(op), dat < nu) = true] → send(RP, DB, uitgevoerd(op, dat)) ·
            RemplacePlanning2(tot, horizon)
         + [and(dat > tijd(op), dat < nu) = false] → foutmelding(tijdstip-verdacht(dat)) ·
            RemplacePlanning(tot, horizon))
    + [correctief(op) = false] →
        sum(gbs in Gebeurtenissen,
              read(RP, DB, gbs) ·
              (   [and(dat ≤ nu, inmarge(dat, op, gbs)) = true] →
                    send(RP, DB, uitgevoerd(op, dat)) ·
                  RemplacePlanning2(tot, horizon)
               + [and(dat ≤ nu, inmarge(dat, op, gbs)) = false] →
                    foutmelding(tijdstip-verdacht(dat)) ·
                  RemplacePlanning(tot, horizon)))

  RemplacePlanning2(tot, horizon) =
    sum(opn in Opdrachten, read(RP, DB, opn) · send(RP, IF, opn)) ·
    RemplacePlanning(tot, horizon)
end RemplacePlanning

process module BeslissingsOndersteuning
begin
  exports
  begin
    processes BeslissingsOndersteuning
  end
  imports Acties-en-Communicatie[87]
  processes
    BeslissingsOndersteuning1 : VliId # LichtpuntIds # Datum
    BeslissingsOndersteuning2 : Vlis # LichtpuntIds # Datum
  variables
    vid :→ VliId
    dat :→ Datum
    lids :→ LichtpuntIds
```

vli : → *Vli*
vlis : → *Vlis*

definitions

BeslissingsOndersteuning =
sum(*vid* **in** *VliId-f*,
sum(*lids* **in** *LichtpuntIds-f*,
sum(*dat* **in** *Datum-f*,
read(*BO*, *IF*, *defect*(*vid*, *lids*, *dat*)) ·
sum(*nu* **in** *Datum-f*,
read(*BO*, *KL*, *tijd*(*nu*)) ·
([(*dat* ≤ *nu*) = *true*] → *BeslissingsOndersteuning1*(*vid*, *lids*, *dat*)
+ [(*dat* ≤ *nu*) = *false*] →
foutmelding(*defect-gedetecteerd-in-toekomst*)))))) ·
BeslissingsOndersteuning

BeslissingsOndersteuning1(*vid*, *lids*, *dat*) =
send(*BO*, *DB*, *start-of-stop*(*vid*)) ·
(**sum**(*vli* **in** *Vli-f*,
read(*BO*, *DB*, *start*(*vli*)) ·
([(*dat* > *tijd*(*vli*)) = *true*] →
([*subset*(*lids*, *lids*(*vli*)) = *true*] → *send*(*BO*, *DB*, *defect*(*vid*, *lids*, *dat*)) ·
send(*BO*, *DB*, *bepaal-actie*(*vid*, *lids*, *dat*))
+ [*subset*(*lids*, *lids*(*vli*)) = *false*] → *foutmelding*(*onbekend*(*lids*)))
+ [(*dat* > *tijd*(*vli*)) = *false*] → *foutmelding*(*niet-onder-schema-op-tijdstip*(*vid*, *dat*))))
+ **sum**(*dat'* **in** *Datum-f*,
read(*BO*, *DB*, *stop*(*vid*, *dat'*)) · *send*(*BO*, *DB*, *onder-schema*(*vid*, *dat*)) ·
(*read*(*BO*, *DB*, *true*) · *send*(*BO*, *DB*, *geef-vli*(*vid*)) ·
sum(*vlis* **in** *Vlis*, *read*(*BO*, *DB*, *vlis*) · *BeslissingsOndersteuning2*(*vlis*, *lids*, *dat*))
+ *read*(*BO*, *DB*, *false*) · *foutmelding*(*niet-onder-schema-op-tijdstip*(*vid*, *dat*))))
+ *read*(*BO*, *DB*, *triviaal-bericht*) · *foutmelding*(*onbekend*(*vid*)))

BeslissingsOndersteuning2(*vli* ∼ *vlis*, *lids*, *dat*) =
[*subset*(*lids*, *lids*(*vli*)) = *true*] → *send*(*BO*, *DB*, *defect*(*vid*(*vli*), *lids*, *dat*)) ·
send(*BO*, *DB*, *bepaal-actie*(*vid*(*vli*), *lids*, *dat*))
+ [*subset*(*lids*, *lids*(*vli*)) = *false*] → *foutmelding*(*onbekend*(*lids*))

end *BeslissingsOndersteuning*

process module *Statistische-Analyse*
begin
exports
begin
atoms
lees-gegevens : *VliId*
analyseer-gegevens : *VliId*
toon-resultaten
processes *Statistische-Analyse*
end
imports *Acties-en-Communicatie*[87]
definitions
Statistische-Analyse =
sum(*vid* **in** *VliId-f*,
read(*SA*, *IF*, *vid*) · *lees-gegevens*(*vid*) · *analyseer-gegevens*(*vid*) · *toon-resultaten*) ·
Statistische-Analyse

end *Statistische-Analyse*

process module *LRproces*
begin
  exports
  begin
    atoms *invoer* : *Duur1*
    processes *BOS*
  end
  imports
    BeslissingsOndersteuning[95], *Configuratie-Analyse*[91], *Database*[90], *Interface*[89], *Klok*[89],
    RemplacePlanning[94], *SchemaBeheer*[93], *Statistische-Analyse*[96]
  definitions
  BOS =
    sum(*dat* **in** *Datum-f*,
        invoer(*dat*) ·
        sum(*Horizon* **in** *Duur1-f*,
            invoer(*Horizon*) ·
            encaps(*H*,
                Klok(*dat*) || *ConfiguratieAnalyse* || *Optimalisator* || *Statistische-Analyse* ||
                Interface || *Database*(*0opn*, *0gbs*) ||
                RemplacePlanning(*dat* + *Horizon* + *een*, *Horizon*) || *SchemaBeheer* ||
                BeslissingsOndersteuning)))
end *LRproces*