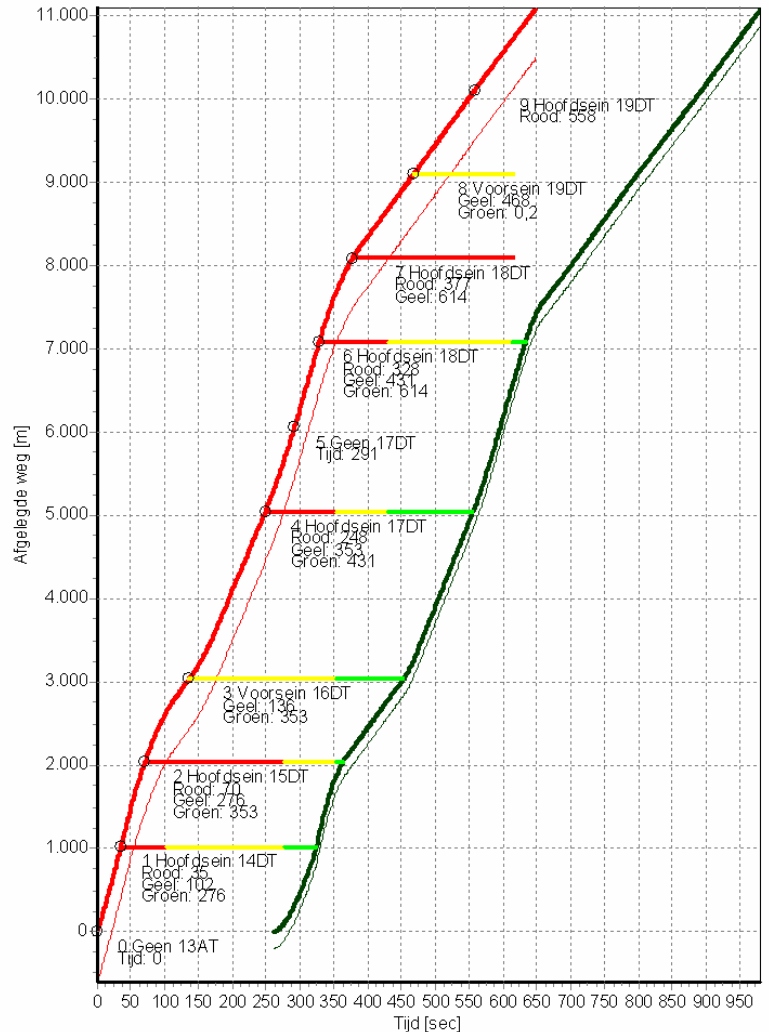


Test Traject voor Verebus TOS
 Trein 1: Freight (600m)
 Trein 2: IRM3 (200m)

```

else
{
  RTGlobalAfgelegdVector2.push_back(RTGlobalAfgele;
  RTGlobalTijdVector2.push_back(RTGlobalTijd2);
}
RTGlobalAfgelegd2 = RTGlobalAfgelegd2 + vtussen;
RTGlobalTijd2 = RTGlobalTijd2 + resolutie;
// Deze waarden zijn alleen BINNEN dit blok
stussen = stussen + vtussen;
tijd = tijd + resolutie;
) while ( (stussen < s ) && ( (vbegin > veind) && (vtussen > 0.0.
// GlobalTijd en GlobalAfgelegd bevatten de tijd en afstand na het af.
// binnen het blok bijgehouden.
if (treinnr == 1)
{
  GlobalTijd = GlobalTijd + tijd;
  
```



Revisie 1.0

Revisie 1.0 - Final
 0.3 - Flow Charts
 0.2 - XML Specificatie

Opdrachtgever:

Verebus Engineering
 Visseringlaan 19c
 2280 CA Rijswijk
 Tel: +31(0)70-3528200
 Fax: +31(0)70-3528205
 Internet: www.veribus.nl

Uitvoering:

Haagse Hogeschool MESO
 Ontwerp: Barry de Graaff

Contactpersoon:
 J. de Korte
 J.deKorte@hhs.nl

Inhoudsopgave

Inhoudsopgave	2
Voorwoord	3
1 Aanleiding van de opdracht	4
2 Opdrachtomschrijving	4
3 Gebruik van TOS	5
3.1 Beschrijving van invoervelden (GUI Frontend)	6
3.2 GUI tips voor GUI Frontend	7
3.3 Traject geldigheid en opmerkingen	8
4 Definities	9
4.1 Baanvakken, blokken en secties	9
4.2 Seinen en snelheidsbeperkingen	10
4.3 Versnellen en vertragen van treinen	12
4.4 Opvolgtijd en zichtafstand	13
5 Berekening in Verebus TOS	15
5.1 Theorie	15
5.2 Implementatie	17
5.3 Berekening van een blok	18
5.4 Berekening van een traject	19
5.5 Berekening van de opvolgtijd	20
5.6 Voorbeeld invoer en uitkomst	23
6 UML Klassendiagrammen en flowcharts	25
7 TOS XML Specificatie	33
7.1 XML Character encoding	33
7.2 StatischTrein.xml	34
7.3 StatischZichtafstand.xml	35
7.4 Traject.xml	36
7.5 Treinen.xml	38
8 Advies voor implementatie van haltes	39
Slotwoord	41
Bijlagen	42

Voorwoord

Voor u ligt de documentatie van het Verebus Trein Opvolgtijden Systeem (TOS) dat is uitgevoerd door Barry de Graaff. Barry is als student verbonden aan de afdeling Micro Elektronica en Systeem Ontwerp van de opleiding Elektrotechniek van de Haagse Hogeschool.

Voor de aanvang van dit project had ik voornamelijk ervaring met het ontwerpen van hardware en embedded (low level) software. Om wat meer kennis op te doen met software op pc-applicatie nivo, heb ik gekozen voor dit project om mee af te studeren. Voor mijn afstuderen fungeert deze documentatie ook als 'scriptie'.

In deze documentatie leg ik eerst kort de voorgeschiedenis en de opdracht van Verebus uit. Meteen na de opdrachtomschrijving vind u een beknopte gebruikershandleiding voor het gebruik van TOS. Hierna worden begrippen toegelicht die in de spoorweg wereld worden gebruikt bij het beschrijven van de problemen/elementen in dit project.

Nadat de begrippen zijn toegelicht, is er een apart hoofdstuk gewijd aan welke berekening plaats vindt in TOS. Na het geven van een voorbeeld is er een hoofdstuk met daarin UML-diagrammen en flowcharts. Deze geven een beeld van hoe de berekening is *geïmplementeerd* in TOS.

Tot slot is er een XML specificatie en een advies voor de implementaties van haltes in een vervolgproject.

Namens MESO wens ik u veel succes toe bij het verder ontwikkelen en gebruiken van Verebus TOS. Voor vragen of informatie over vervolgprojecten kunt u zich wenden tot Dhr. W.J.T. de Kaper (W.J.T.deKaper@hhs.nl).

Barry de Graaff



1 Aanleiding van de opdracht

Verebus is een ingenieursbureau met o.a. defensie en railinfra als grote klanten. Dit project is een opdracht voor de railinfra afdeling van Verebus. Binnen de railmarkt realiseert en adviseert Verebus volledige projecten voor ombouw, sanering, onderhoud en nieuwbouw (functiewijziging en functieherstel) van het spoorwegnet.

Wanneer aanpassingen aan het huidige spoornet uitgevoerd dienen te worden is het noodzakelijk van te voren te onderzoeken wat de gevolgen zijn voor de opvolgtijden van de treinen.

Als we een punt kiezen langs het spoor, dan is de **opvolgtijd** de tijd die verstrijkt tussen het passeren van twee treinen.

Voordat het ontwerp van een nieuwe situatie wordt voorgelegd aan ProRail zullen eerst in een experimentele fase meerder alternatieven onderzocht moeten worden. Deze alternatieven worden uitgewerkt in de vorm van tekeningen van baanvakken.

Voorheen stuurde Verebus deze baanvakken / alternatieven naar een extern bedrijf dat de alternatieven doorrekende qua opvolgtijd. Voor deze berekening kreeg Verebus een factuur.

Doordat het bepalen van een nieuwe situatie in het spoorwegnet een experimenteel karakter heeft is het een kostbare zaak om telkens berekeningen te vragen van een extern bedrijf. En dit is tevens tijdrovend.

2 Opdrachtomschrijving

Verebus wil in dit project een software pakket ontwikkeld hebben waarmee het mogelijk wordt op een **eenvoudige** en **snelle** manier een indicatie te krijgen van de opvolgtijden op een ingevoerd traject.

De nadruk van dit project ligt in de presentatie van de resultaten. Verebus wil een grafiek met op de assen tijd en afstand met daarin plots van twee verschillende treinen. Tussen de plots staan lijnstukken die met kleuren de seinbeelden weergeven. (op een bepaalde afstand en voor een bepaalde tijd)

Verebus heeft als eis dat het programma is geschreven in Builder C++.

Voor een uitgebreide opdrachtomschrijving verwijs ik u naar:

- Startdocument Verebus TOS definitief 0.0.4b.pdf
- Voorstel voor aflevering Verebus TOS.pdf

In de bijlage.

3 Gebruik van TOS

Dit hoofdstuk is een beknopte handleiding voor het gebruik van Verebus TOS. Basis computerkennis en kennis van spoorwegen worden hier verondersteld.

In voorgaande (test) versies was het van belang dat u de landeninstellingen van uw computer instelde. De landeninstellingen / regional and language options zijn niet meer van invloed op de werking van TOS bij alle versies hoger of gelijk aan:

build 070710,
UML 0.0.6,
Codename Meera,
Testversie 2

Al deze en hogere versies van TOS zijn onafhankelijk van uw systeeminstellingen.

3.1 Beschrijving van invoervelden (GUI Frontend)

Traject informatie

Traject titel:

Traject opmerking:

Resolutie: (aantal sec/berekening)

Gegevens van de treinen

Treintype trein 1: Treinlengte trein 1 (m): Beginsnelheid trein 1 (km/h):

Treintype trein 2: Treinlengte trein 2 (m): Beginsnelheid trein 2 (km/h):

Traject blokken

Naam	Type	Lengte (m)	Snelheid groen (km/h)	Snelheid geel (km/h)
13AT	Geen	1000	80	80
14DT	Hoofdsein	1000	100	60
15DT	Hoofdsein	1000	80	60

Blok verwijderen

Nieuw blok toevoegen / bestaand blok wijzigen

Key	Value
Naam	14DT
Type	Hoofdsein
Lengte (m)	1000
Snelheid groen (km/h)	160
Snelheid geel (km/h)	80

Veld	Invoer type		Beschrijving
Traject titel	string	optioneel	Titel van het traject
Traject opmerking	string	optioneel	Opmerking over het traject
Resolutie	double	verplicht	1 / 0,1 / 0,01 seconden per berekening
Beginsnelheid Trein 1 en 2	int	verplicht	beginsnelheid in km/h trein 1 en 2 0 t/m vmax
Treintype Trein 1 en 2	option	verplicht	treintype selecteren (options afkomstig uit StatischTrein.xml)
Treinlengte Trein 1 en 2	int	verplicht	Treinlengte in meters
Blok->Naam	string	optioneel	Bijvoorbeeld 13AT (Bloknaam)
Blok->Type	option	verplicht	Snelheidsbord / Hoofdsein / Voorsein / Halte
Blok->Lengte	int	verplicht	Bloklengte in m
Blok->Snel. groen	int	verplicht	Bloksnelheid in km/h bij een groen sein
Blok->Snel. Geel	int	optioneel	(niet geïmplementeerd in TOS)

3.2 GUI tips voor GUI Frontend

Blokken verplaatsen

In het frame *Traject blokken*, kunt u blokken verplaatsen door erop te klikken en ALT-UP of ALT-DOWN te drukken.

Blokken toevoegen

U kunt een blok wijzigen door op het betreffende blok te klikken in het frame *Traject blokken* en vervolgens in het frame *Nieuw blok toevoegen / bestaand blok wijzigen* de eigenschappen te wijzigen. Vergeet niet hierna op de knop *Blok invoegen* te klikken.

Blok kopiëren

U kunt een blok kopiëren door te klikken op het blok dat u wilt kopiëren. Vervolgens klikt u op een lege regel. Klik nu op de knop *Blok invoegen* in het frame *Nieuw blok toevoegen / bestaand blok wijzigen*. N.b. de onderste regel in *Traject blokken* is altijd leeg.

Blok verwijderen

U kunt een blok verwijderen door te klikken op het blok dat u wilt verwijderen. Vervolgens klikt u op de knop *Blok verwijderen*.

Nieuw blok toevoegen

Klik op een lege regel in het frame *Traject blokken*. Vul de gegevens van het nieuwe blok in, in het frame *Nieuw blok toevoegen / bestaand blok wijzigen*. Vergeet niet hierna op de knop *Blok invoegen* te klikken.

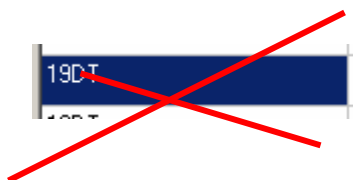
Indien u een blok geselecteerd heeft in frame *Traject blokken*, kunt u ook op enter drukken om een blok in te voegen.

Het is dus essentieel om eerst een regel aan te klikken (selectie) in het frame Traject blokken, voordat u een wijziging uitvoert.

De stippellijn geeft een **selectie** aan:



Dit is geen selectie:



3.3 Traject geldigheid en opmerkingen

- Een traject is ONGELDIG als de lengte van de laatste 2 blokken < treinlengte
- Er moeten ten minste 3 hoofdseinen in een traject zijn.
- Indien er een voorsein in het traject is, moet hier altijd nog minstens 1 hoofdsein na volgen

Traject opmerkingen

- Het achterwiel van trein 1 rijdt alleen het laatste blok binnen als het blok langer is dan de trein.
- De grafiek van trein 2 is *juist* tot en met het laatste sein waarvan de groentijd is bekend. Hierna rijdt deze trein door alsof alle seinen op groen staan.

4 Definities

In dit hoofdstuk worden definities uitgelegd die voortkomen uit de railwereld. Ook worden er begrippen uitgelegd die specifiek van toepassing zijn op de opvolgtijd. Het begrijpen van deze begrippen is nodig om in het volgende hoofdstuk te kunnen lezen hoe TOS het traject en de opvolgtijd berekend.

4.1 Baanvakken, blokken en secties

Een **baanvak** is het grootste stuk spoor, het kan alles zijn tussen twee punten A en B. Voorbeeld: Baanvak Leiden Centraal – Schiphol

Maar het hoeven geen stations te zijn. Het is een vrije definitie. Denk eraan dat een baanvak ook parallel sporen bevat. (dus meerdere sporen) Er kunnen zich dus meerdere treinen in een baanvak op hetzelfde moment bevinden.

Blok



Een blok is een stuk spoor dat wordt begrensd door seinen of een snelheidsbord, **binnen een blok geldt overal dezelfde snelheid**, MITS er een voorsein in het staat. (zie verder) Bij de seinen staat een verticale streep in het spoor getekend. Dit is een las, deze vormt een isolatie in het spoor.

In tekeningen hebben **blokken** cijfers (14AT)

Sectie




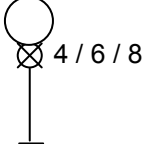


Als er een trein op het spoor staat maakt deze 'kortsluiting' tussen de rails. Hierdoor komt een blok op bezet te staan. Dit werkt met relais. (Kortsluitlans principe)

De belangrijkste reden voor het maken van secties is een elektrische afweging. Zodat de detectie van de trein 'de kortsluiting' kan worden gewaarborgd.

Ook overwegen, bruggen e.d. zijn een reden voor het maken van secties. Een sectie is in principe het kleinste element in het spoor.

In tekeningen hebben **secties** letters (14CT)

4.2 Seinen en snelheidsbeperkingen

Hoofdsein	Hoofdsein met lichtbak	Voorsein	Dwergsein	
				

Hoofdsein of hoofdsein met lichtbak

Een hoofdsein, geeft informatie over de SNELHEID maar ook over de BEZETTING van het spoor. (Is er een trein in een blok)

Staat een hoofdsein op rood (stand stop) dan staat het sein ervoor op geel en het sein daarvoor op groen. Het sein heeft dus een relatie met het sein ervoor.

Een trein die in een blok 120 km/h mag rijden (volgens het seinbeeld) komt NOOIT een rood sein tegen. Seinen gaan voor borden.

- **Groen** : volgende sein is uit de stand stop.
- **Groen knipperend** : passeren met max. 40 km/uur. Daarvoor staat het sein dan meestal geel + 4.
- **Groen knipperend met een matrix cijfer (lichtbak) eronder** : sein passeren met de voorgeschreven snelheid. (bijv. 6 = 60 km/uur), volgende sein is uit de stand stop.(Zie voorbeeld sein).
- **Geel** : snelheid verminderen en rekenen op stop. Meestal staat dan het volgende sein rood.
- **Geel met een matrix cijfer eronder** : snelheid verminderen tot voorgeschreven snelheid (bijv. 8 = 80 km/uur). Bij het volgende sein moet de opgelegde snelheid zijn bereikt. Het volgende sein zal geen stop vertonen.
- **Geel met knipperend matrix cijfer eronder** : snelheid verminderen tot voorgeschreven snelheid (bijv. 6= 60 km/uur). Bij het volgende sein hoeft de snelheid nog niet bereikt te zijn. Machinist moet echter blijven remmen tot de getoonde snelheid bereikt is. Het volgende sein zal geen stop vertonen.
- **Geel knipperend** : rijden op zicht (ROZ) en kunnen stoppen voor elk gevaar (bijv.: als er gekoppeld wordt).
- **Rood** : stop voor het sein.

Snelheden zijn altijd uitgedrukt in tientallen, dus 8 = 80 km/h, 10 = 100 km/h enz.

Voorsein

Een voorsein, geeft informatie over de SNELHEID. Maar niet of er een trein in een bepaald blok is.

Een voorsein heeft een relatie met het hoofdsein **erna**. Anders gezegd een voorsein gaat vooraf aan een hoofdsein. Een voorsein kan geen rood vertonen.

Dwergsein

Indien groen passeren met max. 40 km/uur.



figuur 1a, hoofdsein figuur 1b, dwergsein

Bron: Max Hovens <http://www.treinen.demon.nl>

Begrenzingsborden

Snelheidsverminderingbord	Baanvak Snelheidsbord	Snelheidsvermeerderingsbord
		

Bron: Max Hovens <http://www.treinen.demon.nl>

Er zijn drie borden om de snelheid aan te geven. Het snelheidsverminderingbord betekend de snelheid begrenzen tot door het getal aangegeven snelheid.

Het baanvak snelheidsbord en het snelheidsvermeerderingsbord zijn hetzelfde. ProRail heeft aangegeven dat het snelheidsvermeerderingsbord niet meer wordt geplaatst. Daar waar het nog staat laat men het staan tot er aanpassingen plaats vinden aan het spoor.

Dit betekend dat beide borden moeten worden geïnterpreteerd als snelheid verhogen tot de door het getal aangegeven snelheid.

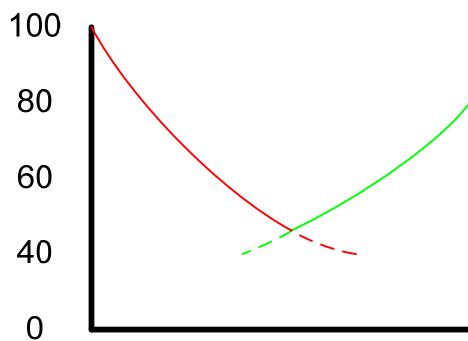
4.3 Versnellen en vertragen van treinen

Treinen vertragen en versnellen bij snelheidswijzigingen maximaal, volgens de acceleratie en deceleratie constanten van de trein (StatischTrein.xml zie verder).

Tussentijdse snelheidswijziging

Bij remmingen van bijv. 120 naar 100 door een beperkend seinbeeld t.g.v. bijv. een wissel dient eveneens de locatie van de las als uitgangspunt te worden gebruikt. Dit i.v.m. de ATB code. Direct vanaf de las dient de remming ingezet te worden met de vertraging volgens de tabel. Totdat de geëiste snelheid is bereikt.

Mits tussentijds deze snelheidseis weer is aangepast. Het kan namelijk dan gedurende de remming de trein vanaf een bepaald punt weer mag versnellen. De remming wordt dan gestaakt en direct wordt de versnelling weer ingezet. Zo zal de remming in dit geval dus niet voltooid worden.



Hierboven daar een voorbeeld van. De trein rijdt 100 en remt af naar 40 km/h. Tijdens deze remming krijgt hij een teken dat hij 80 mag gaan rijden. Hij staakt de remming en zet de versnelling weer in tot hij de 80km/h heeft bereikt. (de kleuren in de grafiek zijn ter indicatie van een remming of versnelling, dit staat even los van de seinbeelden)

4.4 Opvolgtijd en zichtafstand

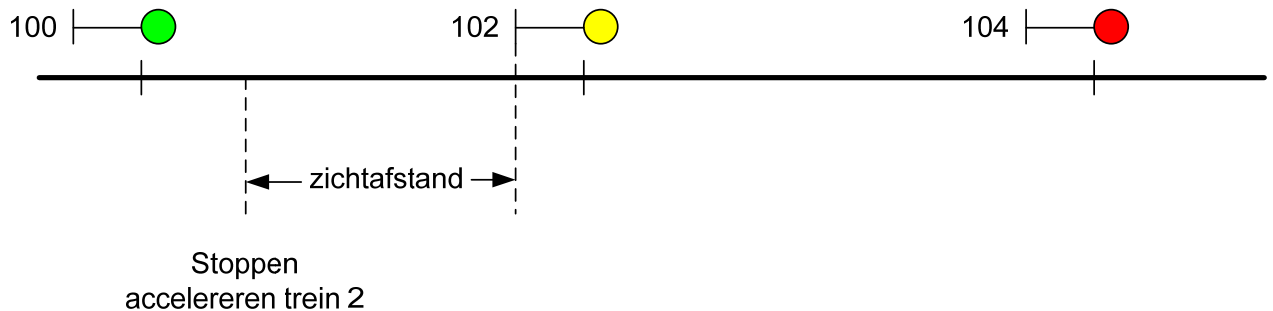
Opvolgtijd

Als we een punt kiezen langs het spoor, dan is de **opvolgtijd** de tijd die verstrijkt tussen het passeren van twee treinen.

Ideale opvolgtijd

Trein 2 rijdt achter trein 1 aan, en **ziet** daarbij alleen groene seinen. De tijd dat de seinen al op groen staan na het passeren van trein 1 moet zo kort mogelijk zijn. Hiervoor is de zichtafstand van belang.

Zichtafstand



Wat betreft trein 2 heb je te maken met een belemmering in aanzet. Stel sein 104 staat op rood zoals onderstaand voorbeeld. Sein 102 toont dan geel en sein 100 groen.

Stel dat trein 2 net is vertrokken bij sein 100 en dus vanuit stilstand optrekt. Treinen zijn traag en we gaan er voor het voorbeeld vanuit dat hij zijn gewenste snelheid nog niet heeft bereikt wanneer hij sein 102 nadert. Omdat de machinist geel ziet zal hij zijn acceleratie niet doorzetten. Dit is een ongewenste situatie, de machinist moet te allen tijde door kunnen rijden.

Hierbij is de positie van sein 102 + zichtafstand van belang voor de bepaling op welk tijdstip trein 2 op zijn vroegst achter trein 1 mag rijden. Je moet dus trein 2 later laten vertrekken vanaf sein 100 indien hij eerder op het remmingspunt aanwezig is.

Dit punt is benodigd om te bepalen wat de meest ideale opvolgtijd is.

Zichtafstand (vervolg)

Een machinist ziet een sein een aantal meters voordat de trein het sein passeert.

Afhankelijk van de snelheid van de trein, wordt er een andere zichtafstand gehanteerd. De zichtafstand wordt berekend aan statische gegevens uit StatischZichtafstand.xml.

In StatischZichtafstand.xml staat de zichtafstand in meter aangegeven met een overeenkomstige snelheid km/h. TOS rekent deze gegevens om naar een correctie in seconden.

Voor de zichtafstand is de snelheid van belang in het blok **dat voorafgaat** aan het **kritieke punt** (in principe is de zichtafstand ook van toepassing op de andere seinen, alleen zijn die niet beperkend).

UITZONDERING:

Indien het kritieke punt blok 0 is, dan nemen we de snelheid van blok 0 als referentie voor de zichtafstand

Kritieke punt

Het kritieke punt wordt gedefinieerd als: een locatie in het Traject waar een hoofdsein staat. En waar de tijd tussen het op rood en op groen gaan van het sein langer is dan bij welk ander sein dan ook in het traject.

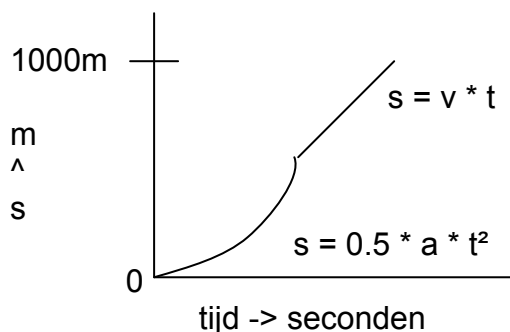
5 Berekening in Verebus TOS

Dit hoofdstuk legt de theorie achter de opvolgtijd uit. Hierna kunt u lezen hoe deze theorie is gebruikt om de treinen en de opvolgtijd uit te rekenen.

Uiteindelijk is het doel van dit hoofdstuk dat u begrijpt in welke volgorde de berekening en dus de functies van TOS worden uitgevoerd.

5.1 Theorie

In deze paragraaf wordt kort de basis natuurkunde uitgelegd die van toepassing is voor het berekenen van de tijd die het kost om een bepaalde afstand af te leggen.



Gegeven:

Een trein trekt vanaf 0 km/h op naar 40 km/h er rijdt dan verder met 40 km/h totdat de trein 1000m heeft afgelegd.

Gevraagd:

Hoe lang doet de trein erover in seconden om 1000m te bereiken als voor dit trientype de versnelling 0,5 m/s² is?

Oplossing:

In het eerste deel van de curve is de trein van 0km/h aan het versnellen naar 40 km/h, de formule $s = 0.5 * a * t^2$ is hier van toepassing. Hierin is s de afgelegde weg in meter, a de versnelling in m/s² en t de tijd in seconden.

We kunnen deze formule nu niet toepassen omdat we s en t niet weten...

De versnelling a, is het verschil van afgelegde weg delen door de tijd, uit de formule $a = dv / dt$ kunnen we nu de tijd uitrekenen:

$$40 \text{ km/h} / 3,6 = 11,11 \text{ m/s}$$

$$a = 11,11 / dt$$

$$0,5 / 1 = 11,11 / dt$$

$$dt = 11,11 / 0,5 = 22,22 \text{ seconden}$$

Hieruit volgt dat de versnelling 22,22 seconden duurt, nu kunnen we de eerdere formule invullen:

$$s = 0.5 * a * t^2$$

$$s = 0,5 * 0,5 * 22,22$$

$$s = 123,43 \text{ m}$$

In het tweede deel van de curve is de snelheid constant 40 km/h. We weten dat het blok 1000 m lang is. Dit betekent dat de lengte die nog moet worden afgelegd in het tweede deel van de curve is:

$$1000 - 123,43 = 876,57 \text{ m}$$

Als we nu invullen

$$s = v * t$$

$$t = s / v$$

$$t = 876,57 / 11,11$$

$$t = 78,90 \text{ seconden}$$

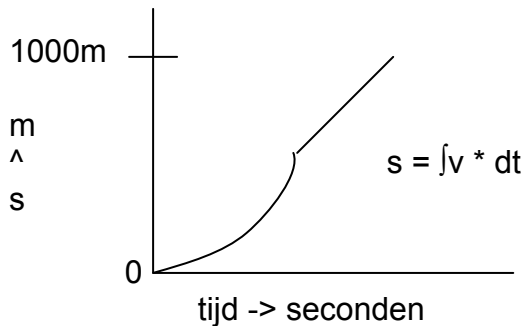
Dan is de tijd om dit blok te doorlopen: **101,12** seconden.

Door gebruik te maken van deze formules kan ook worden gecontroleerd dat de berekeningen van Verebus TOS juist zijn. Echter is dan de tijd (101,12) bekend en moet de afstand worden berekend.

5.2 Implementatie

De methode die is beschreven in paragraaf 5.1 is goed om te controleren dat het antwoord dat TOS geeft juist is. Echter het kan niet worden gebruikt in de software.

De belangrijkste reden waarom deze methode niet kan worden gebruikt is, dat de versnelling a van een trein verandert naarmate deze sneller of langzamer rijdt. Een trein die van 0 km/h gaat versnellen doet dit met een andere snelheid dan een trein die vanaf 100 km/h gaat versnellen.



De oplossing van dit probleem zit in het *benaderen* van de curve met een integraal. Door een *klein interval van t* te nemen en steeds de momentele snelheid te bepalen kunnen we de afgelegde weg berekenen bij iedere waarde van t .

De momentele snelheid v bestaat uit de huidige snelheid in m/s plus voor iedere seconde de eventuele versnelling in m/s^2 . In andere woorden door het kleine interval wordt **een versnelling ook als een lineaire snelheid berekend**.

5.3 Berekening van een blok

TOS maakt alleen gebruik van de basisbewerkingen (optellen, vermenigvuldigen, delen en aftrekken) om de berekening uit 5.2 te doen. Het programma refereert aan het kleine interval met het woord *resolutie*. Resolutie kan men vertalen als het aantal seconden per berekening.

TOS doet een reeks berekeningen voor ieder blok, totdat het einde van het blok is bereikt. Hieronder is beschreven hoe de berekening verloopt die iedere seconde wordt uitgevoerd. (Als de resolutie 1 seconde is)

Voor een trein met een beginsnelheid 0 km/h en een versnelling van 0,5 m/s² komt de berekening er als volgt uit te zien in Verebus TOS.

tijd	a _{huidig}	vtussen	stussen
0	0	0 + 0 = 0	0
1	0,5	0,5 + 0 = 0,5	0,5
2	1	1 + 0,5	1,5
3	1,5	1,5 + 1,5	3
4	2	2 + 3	5
5	2,5	2,5 + 5	7,5
6	3	3 + 7,5	10,5

$$a_{\text{huidig}} = \text{tijd} * 0,5$$

(0,5 constante uit trein tabel, deze tabel laat alleen het begin van de berekening zien)

In TOS moet er ook rekening worden gehouden met resoluties kleiner dan 1. Dit moet in de berekening worden verwerkt. Ook moeten allerlei voorwaarden worden gecontroleerd om de berekening te vervolgen of om deze af te breken.

De basis van de berekening vertalen we in code als:

```
vtussen = vbegin
vtussen = vtussen / 3.6
vtussen = vtussen + tijd * a
vtussen = vtussen * resolutie
```

```
stussen = stussen + vtussen
tijd = tijd + resolutie
```

De voorwaarden van een while lus worden gebruikt om verder te rekenen, of om verder te gaan met een volgende blok.

5.4 Berekening van een traject

Een traject bestaat uit een aantal blokken waarvoor een hoofdsein, voorsein of een snelheidsbord staat. Eigenschappen van een blok zijn:

	data type	beschrijving
Naam	string optioneel	bloknaam bijvoorbeeld 14AT
Type	string verplicht	Hoofdsein, Voorsein of Snelheidsbord
Lengte	integer	Lengte van het blok in meter
Bloksnelheid	integer	Bloksnelheid van trein 1, 2 in km/h

Een traject is geldig, kan dus worden berekenend als:

- De lengte van de laatste 2 blokken < treinlengte.
- Er ten minste 3 hoofdseinen in een traject zijn.
- Indien er een voorsein in het traject is, hier altijd nog minstens 1 hoofdsein na volgt.

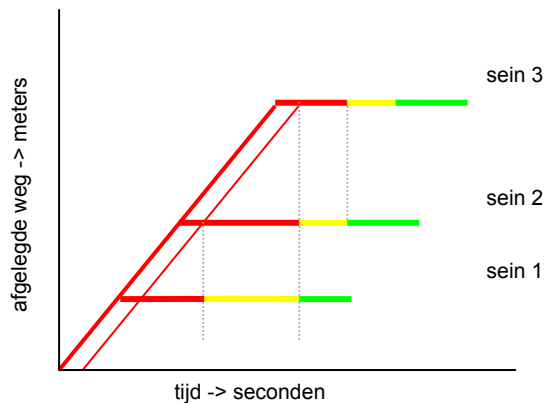
Voor ieder blok worden de gegevens uit XML gelezen door de routine Calculator::BerekenTraject. Deze gegevens worden samen met de waarden uit voorgaande berekeningen doorgegeven aan de routine Calculator::BerekenBlok.

De berekening binnen Calculator::BerekenBlok is zoals beschreven in paragraaf 5.3. Calculator::BerekenBlok is ook verantwoordelijk voor het opslaan van de berekende waarden in het geheugen.

5.5 Berekening van de opvolgtijd

Nadat het traject is berekend voor trein 1, worden de seinbeelden berekend door de routine BerekenSeinbeelden::BerekenSeinbeeldenGeel en BerekenSeinbeelden::BerekenSeinbeeldenGroen.

De situatie zou er dan als volgt uit kunnen zien:



Grafiek 5.1: Trajectberekening na trein 1

De (diagonale) dikke rode lijn in grafiek 5.1 is het voorwiel van de eerste trein die het traject inrijdt (trein 1). De dunne lijn is het achterwiel van trein 1.

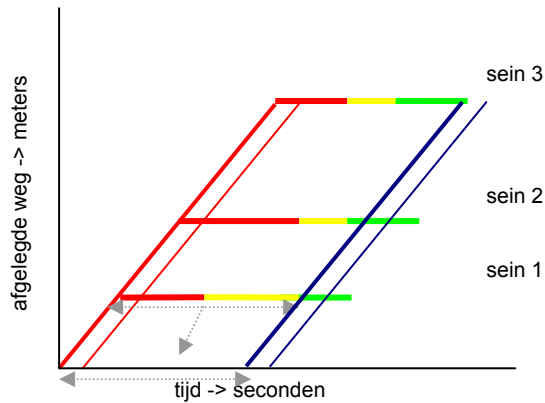
De horizontale rode, gele en groene lijnen symboliseren de seinbeelden. Om het voorbeeld simpel te houden gebruiken we alleen blokken met hoofdseinen. Merk op dat voor sein 3 ook de gele en groene tijden bekend zijn. (Terwijl dit eigenlijk niet kan worden berekend in een traject met 3 blokken.)

Nu berekend TOS welk sein het meest beperkend is voor de opvolgtijd. Dit noemen we het kritieke punt. **Het kritieke punt wordt gedefinieerd als: een locatie in het Traject waar een hoofdsein staat. En waar de tijd tussen het op rood en op groen gaan van het sein langer is dan bij welk ander sein dan ook in het traject.**

In dit voorbeeld is het kritieke punt sein 1.

Nu wordt de routine Calculator::BerekenTraject opnieuw aangeroepen voor trein 2... maar in plaats van dat we trein 2 op $t = 0$ seconden laten vertrekken, laten we trein 2 vertrekken op tijdstip $t = 0 + (t_{\text{groen1}} - t_{\text{rood1}})$.

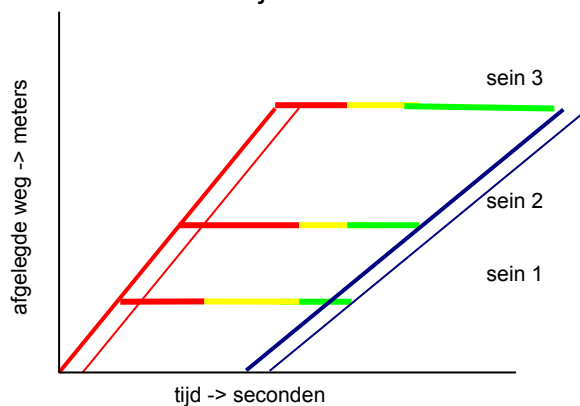
De situatie zou er dan als volgt uit kunnen zien:



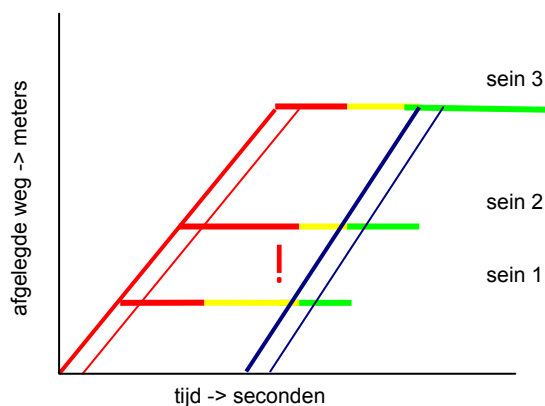
Grafiek 5.2: Trajectberekening na trein 2 met kritiek punt

De grijze horizontale stippellijn laat de offset van het kritieke punt zien.

De blauwe lijn in grafiek 5.2 is trein 2. Indien trein 2 exact hetzelfde type heeft als trein 1, dan zou deze opvolging optimaal zijn. Maar in de praktijk kan trein 2 langzamer of sneller zijn:



Grafiek 5.3: Trajectberekening na langzamere trein 2 met kritiek punt



Grafiek 5.4: Trajectberekening na snellere trein 2 met kritiek punt

In grafiek 5.3 is te zien dat sein 1 al een tijd op groen stond voordat trein 2 passeerde. Dit is niet optimaal omdat trein 2 dus eerder kan vertrekken.

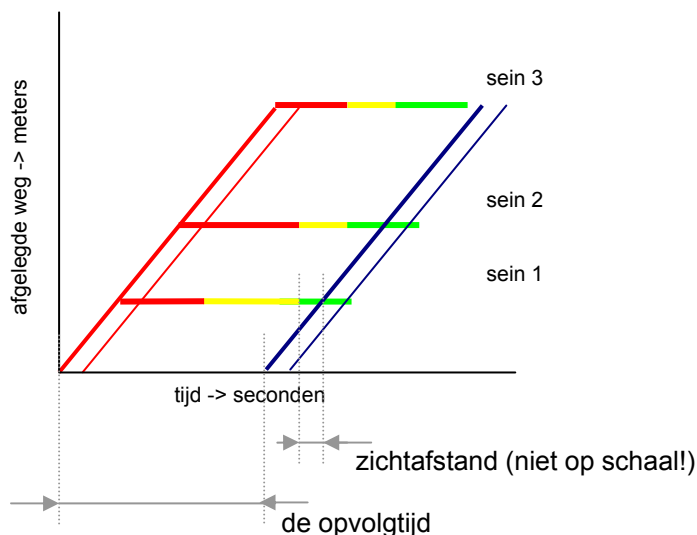
In grafiek 5.4 is te zien dat trein 2 sneller is en dat de machinist een geel sein ziet. Het doel van de opvolgtijdberekening is dat de machinist alleen groen ziet. De situatie in grafiek 5.4 is dus niet gewenst. Trein 2 moet later vertrekken.

TOS berekend hoe groot het tijdsverschil (**correctie** positief of negatief) is in het kritieke punt. Zodat sein 1 zo kort mogelijk op groen staat en de machinist niet geel ziet.

Hier is de zichtafstand (zie paragraaf 4.4) ook van belang. Immers het sein moet al op groen staan zodra de machinist het sein ziet. Niet op het moment dat de trein bij het sein is.

Routine Traject::BerekenZichtafstand: berekening van de zichtafstand in seconden
Routine Traject::BerekenOptimaleOpvolgtijd: berekening van de correctie

De uiteindelijke opvolgtijd ziet er dan zo uit:



Grafiek 5.5: Trajectberekening optimale opvolgtijd

Samenvattend: de optimale opvolgtijd voor trein 2 gaat via een trial en error principe. De routines worden aangeroepen in deze volgorde:

1. Bereken trein 1 (Calculator::BerekenTraject en Calculator::BerekenBlok)
2. Bereken achterwiel trein 1
3. Bereken kritieke punt (Traject::BerekenKritiekPunt)
4. Bereken trein 2 met als offset het kritieke punt
5. Bereken correctie & zichtafstand (Traject::BerekenOptimaleOpvolgtijd Traject::BerekenZichtafstand)
6. Bereken trein 2 nogmaals (met correctie en zichtafstand)

5.6 Voorbeeld invoer en uitkomst

Traject.xml (tekst)	Treinen.xml (tekst)
<p>Traject Title="Test Traject voor Verebus TOS" Remark="Variant: Plaats hier een opmerking over dit traject."</p> <p>Blok Title="13AT" Type Snelheidsbord Lengte 1000 Bloksnelheid 80 Resolutie 0,01</p> <p>Blok Title="14DT" Type Hoofdsein Lengte 1000 Bloksnelheid 100</p> <p>Blok Title="15DT" Type Hoofdsein Lengte 1000 Bloksnelheid 80</p> <p>Blok Title="16DT" Type Voorsein Lengte 2000 Bloksnelheid 70</p> <p>Blok Title="17DT" Type Hoofdsein Lengte 1000 Bloksnelheid 160</p> <p>Blok Title="17DT" Type Snelheidsbord Lengte 1000 Bloksnelheid 100</p> <p>Blok Title="18DT" Type Hoofdsein Lengte 1000 Bloksnelheid 40</p> <p>Blok Title="19DT" Type Voorsein Lengte 1000 Bloksnelheid 40</p> <p>Blok Title="18DT" Type Hoofdsein Lengte 1000 Bloksnelheid 40</p> <p>Blok Title="19DT" Type Hoofdsein Lengte 1000 Bloksnelheid 40</p>	<p>TreinTitle="Trein1:Freight" Type Freight Lengte 600 Beginsnelheid 0</p> <p>TreinTitle="Trein2:DD-AR4" Type DD-AR4 Lengte 200 Beginsnelheid 40</p>

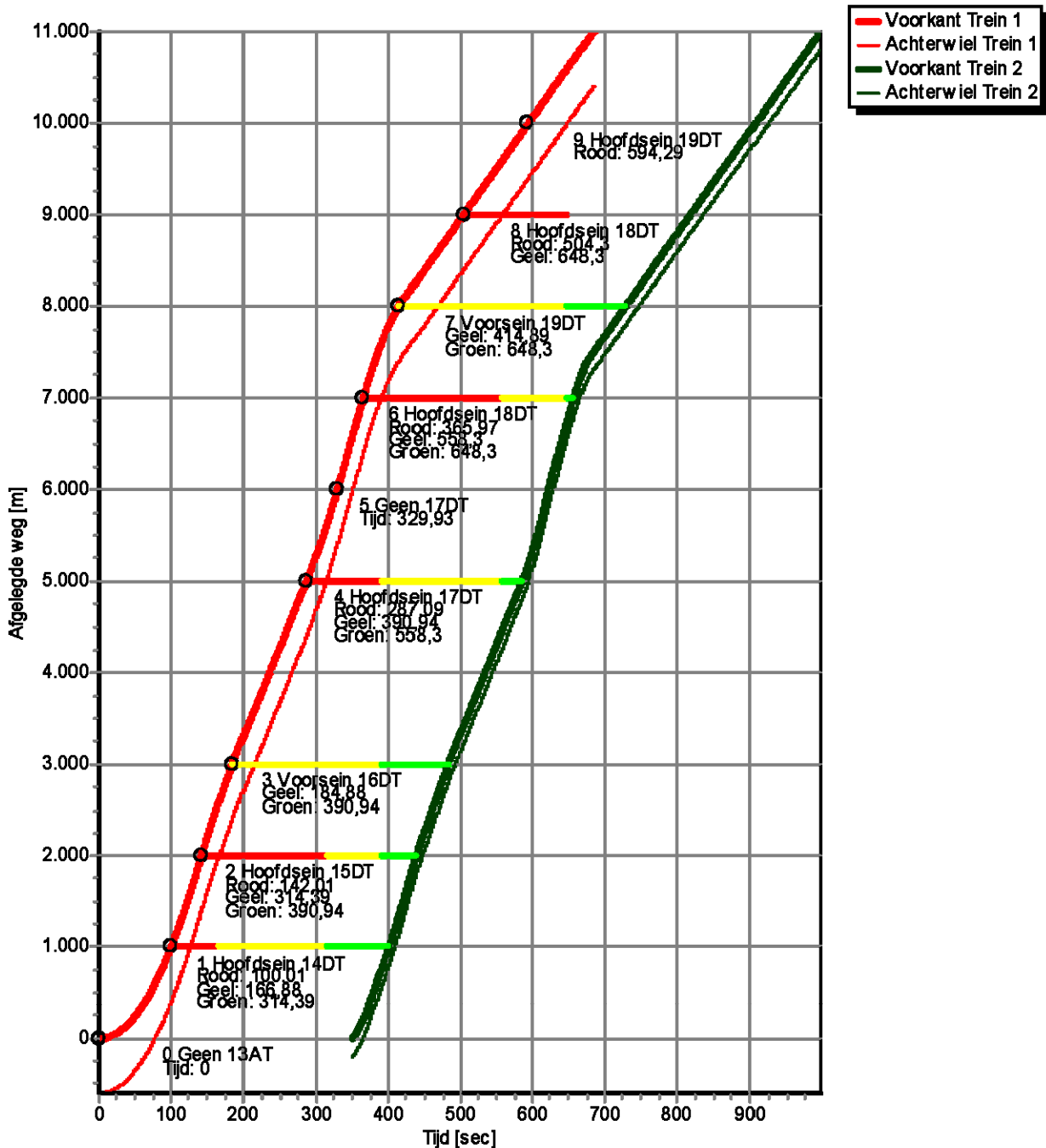
Project: Test Traject voor Verebus TOS

Variant: Plaats hier een opmerking over dit traject.
 Opgeloopte tijd: 351,35 seconden, kritiek blok: 6.

Trein 1
 Materieeltype: Freight
 Treinlengte: 600m

Trein 2
 Materieeltype: DD-AR4
 Treinlengte: 200m

Test Traject voor Verebus TOS
 Trein 1: Freight (600m)
 Trein 2: DD-AR4 (200m)



6 UML Klassendiagrammen en flowcharts

In dit hoofdstuk zijn de UML klassendiagrammen en de flowcharts van Verebus TOS te vinden.

Het is belangrijk om te beseffen dat het project eigenlijk uit 2 programma's bestaat. Namelijk de GUI Frontend TOSFrontend.exe (met alle invoervelden) en de TOS Calculator (met de grafiek) TOS.exe.

De UML klassendiagrammen geven een overzicht van welke klassen er zijn en welke functies/methodes en variabelen beschikbaar zijn in die klasse. Van iedere klasse wordt in dit project maar 1 instantie/object aangemaakt. Daarom is er geen apart object diagram weergegeven.

De flowcharts geven aan hoe het programma verloop eruitziet. De termen die in de flowcharts worden gebruikt spreken voor zich als u eerst het vorige hoofdstuk heeft gelezen. M.a.w de namen van de klassen en methoden geven duidelijk weer wat de bewerking is van die klasse of methode. Soms wordt de werking nog verduidelijkt door een extra opmerking in de flowchart.

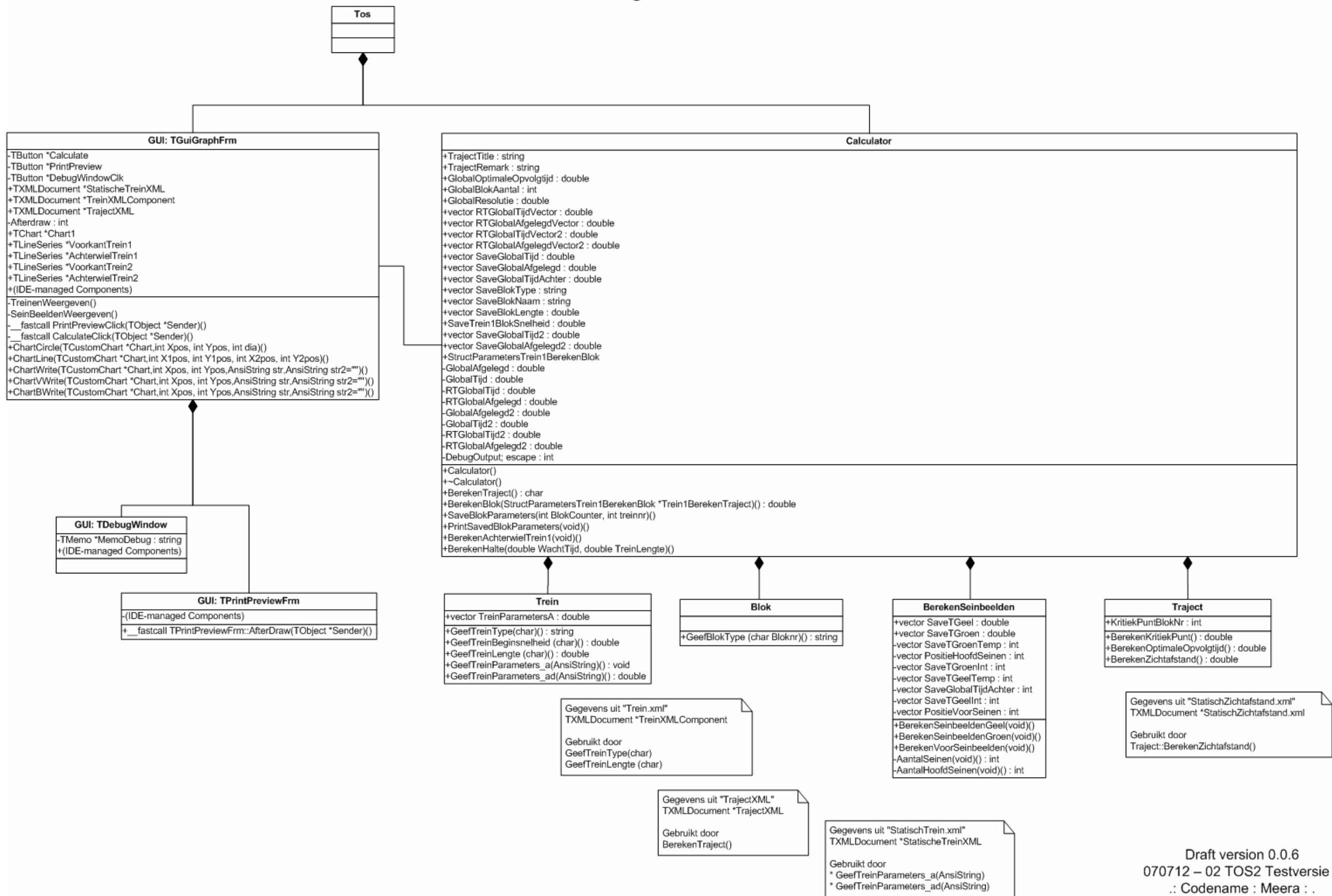
Flowchart hiërarchie

Ter verduidelijking geef ik hier de hiërarchie aan die in de flowcharts is gebruikt. Dit wordt in de flowcharts aangegeven met bijvoorbeeld:



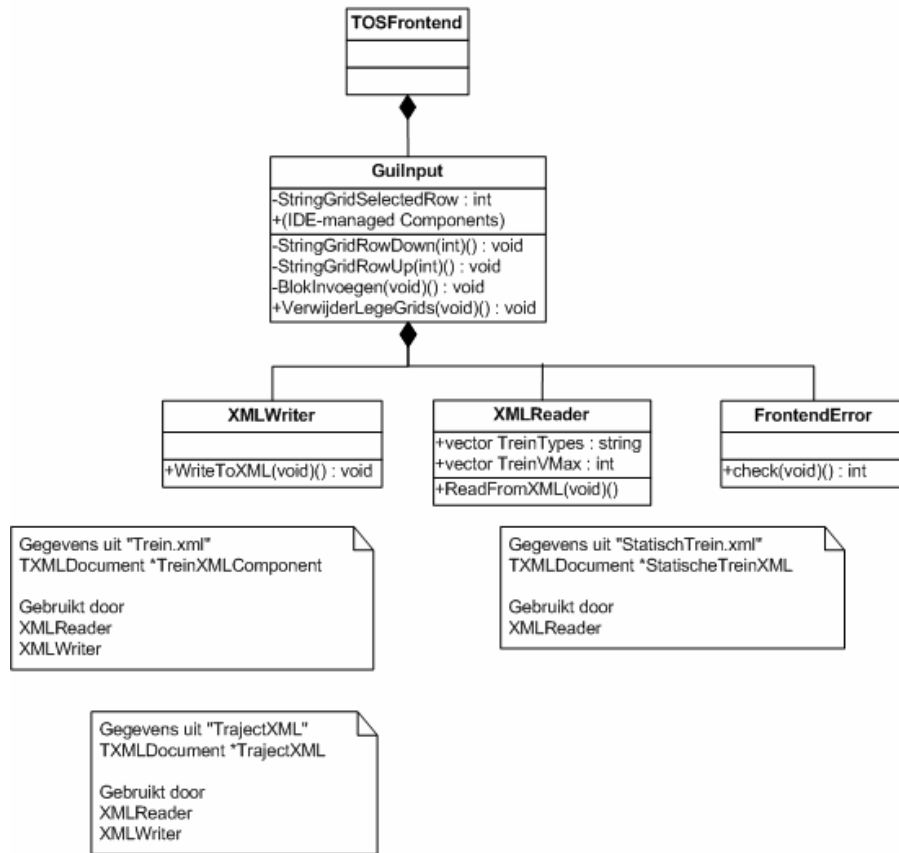
```
-Project Verebus TOS
|
|-> Frontend
|
-> Tos
    |
    |-> Calculator::
    |   |-> BerekenTraject
    |   -> Berekenblok
    -> BerkenSeinbeelden::
        |-> BerekenSeinbeeldenGeel
        |-> BerekenSeinbeeldenGroen
        -> ...
```

Klassendiagram Verebus TOS



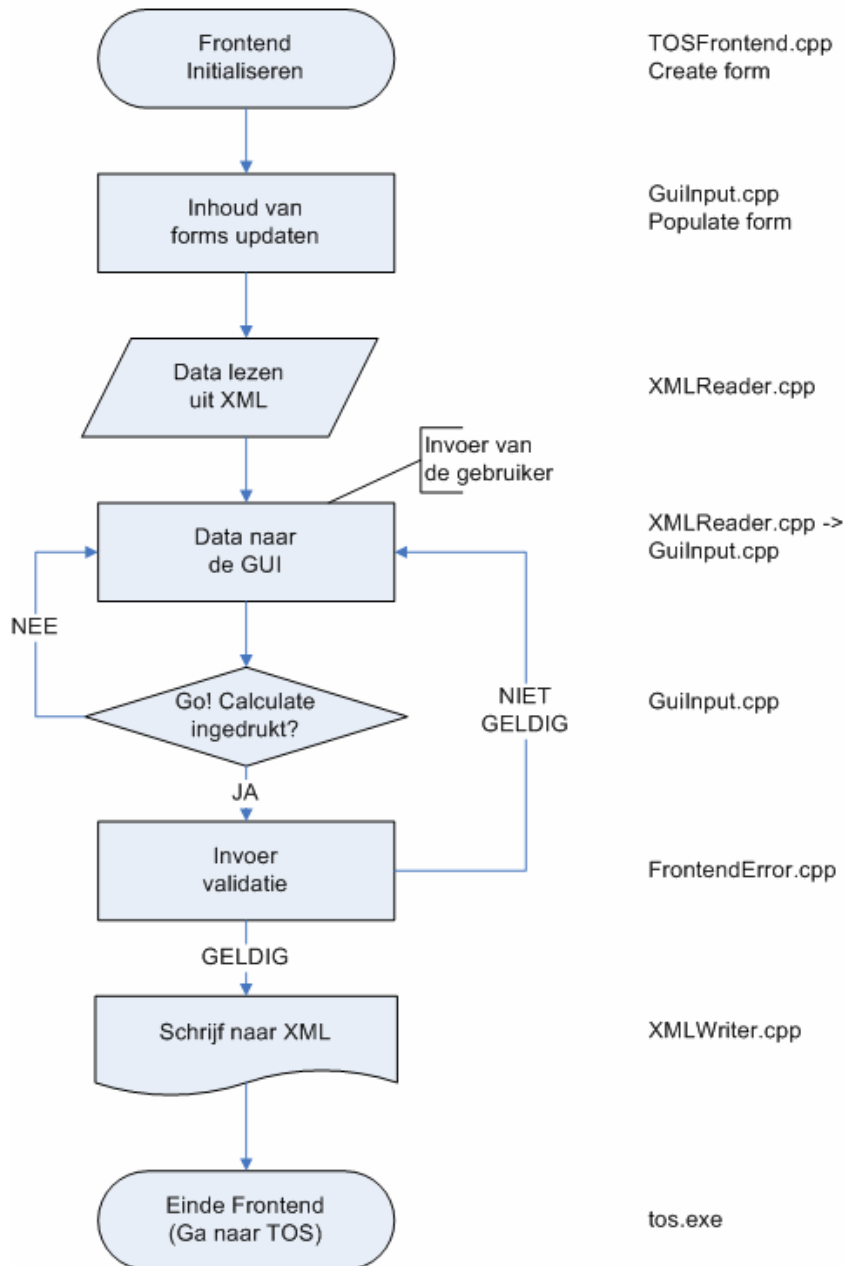
Draft version 0.0.6
070712 – 02 TOS2 Testversie 2
.: Codename : Meera : .

Klassendiagram Verebus TOS Frontend (GUI)



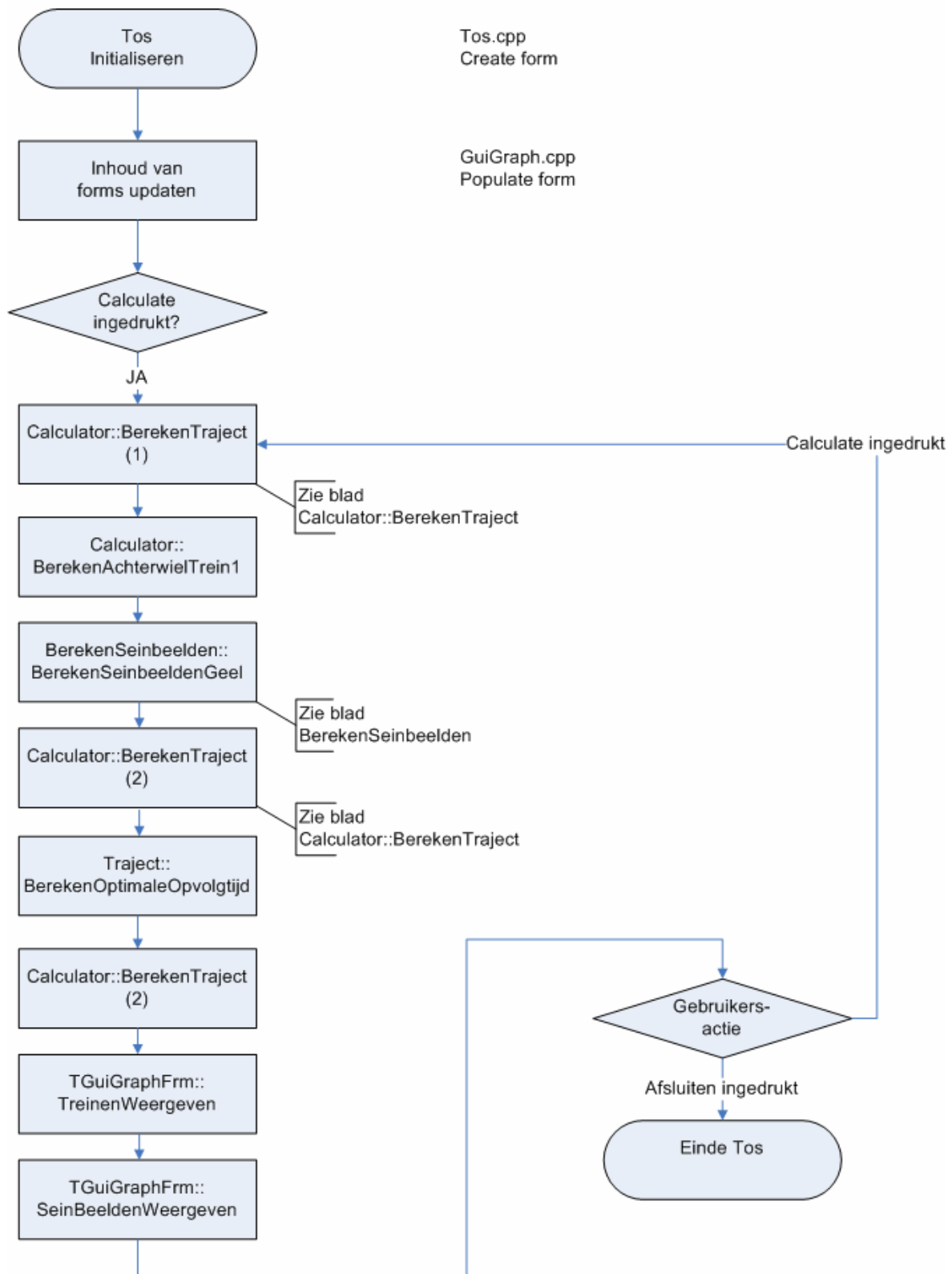
Draft version 0.0.6
070712 – 02 Frontend Testversie 2
For .: Codename : Meera : .

Flowchart Verebus TOS > Frontend



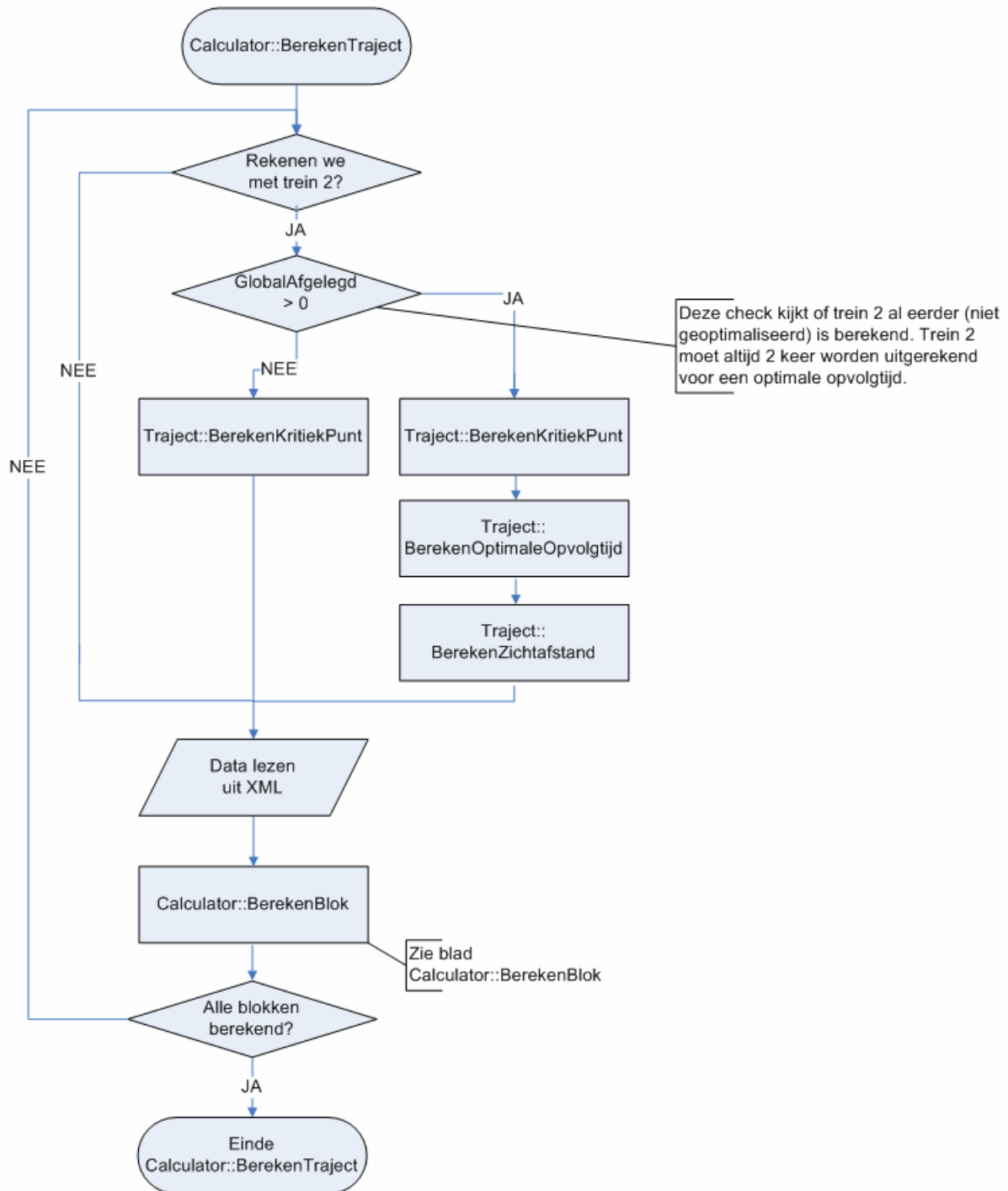
Draft version 0.0.6
 070712 – 02 Frontend Testversie 2
 For .: Codename : Meera : .

Flowchart Verebus TOS > Tos



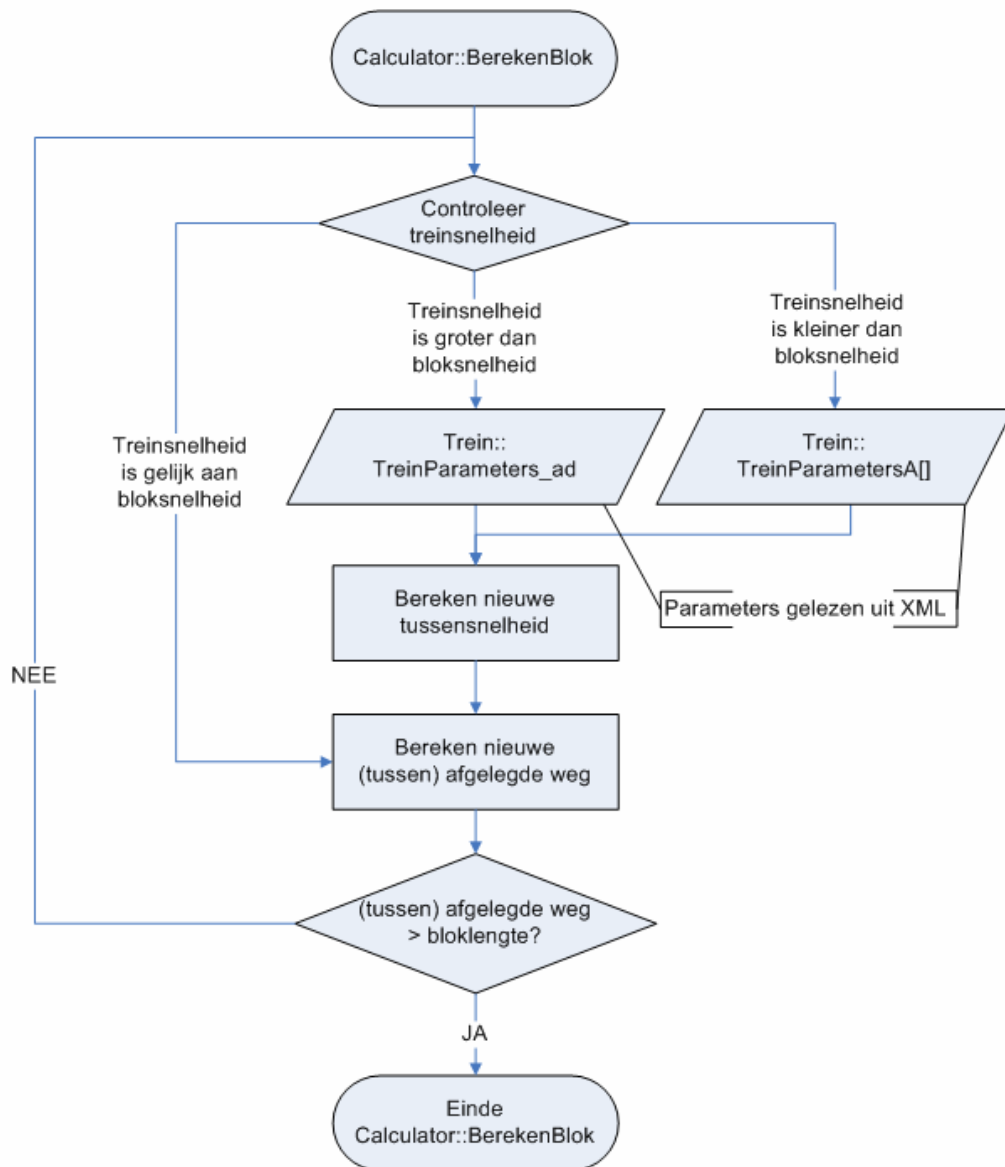
Draft version 0.0.6
070712 – 02 Frontend Testversie 2
For .: Codename : Meera : .

Flowchart Verebus TOS > Calculator::BerekenTraject



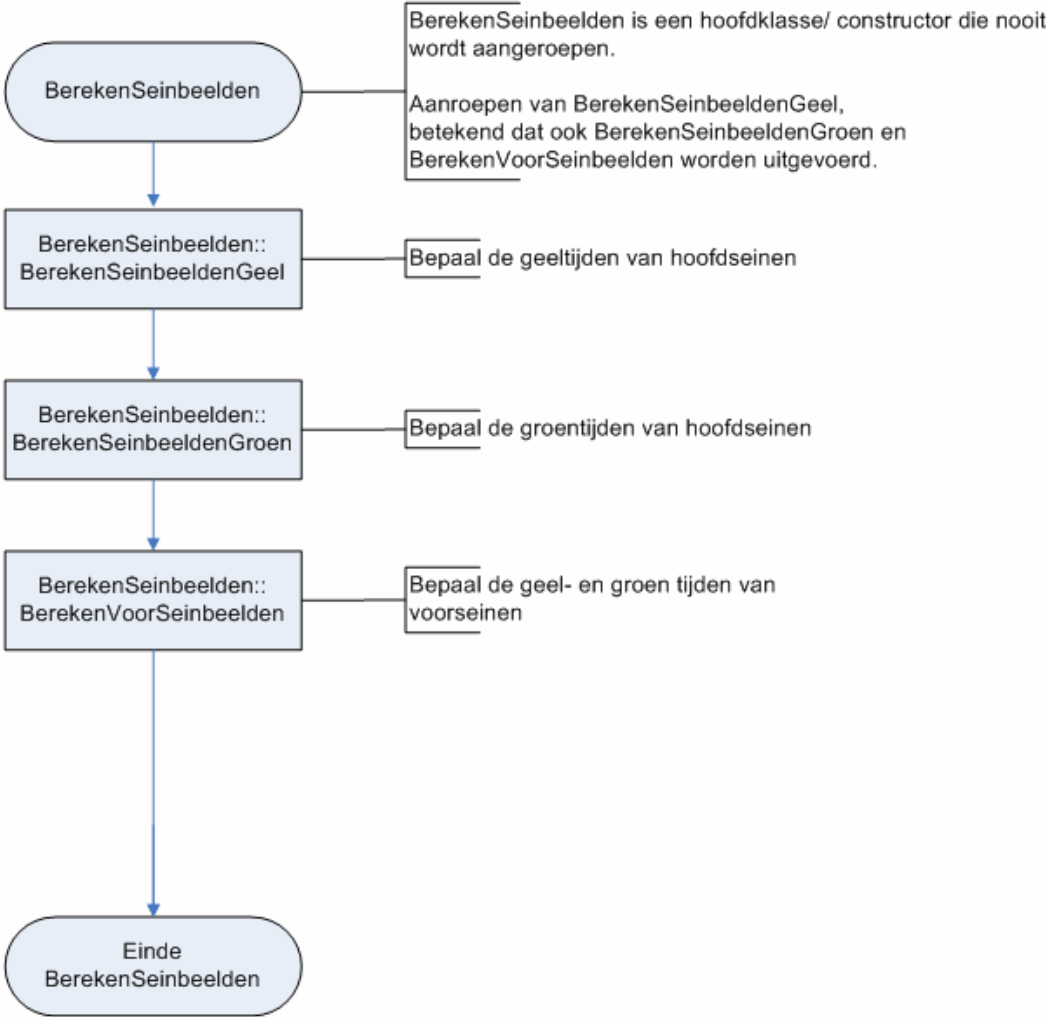
Draft version 0.0.6
 070712 – 02 Frontend Testversie 2
 For .: Codename : Meera : .

Flowchart Verebus TOS > Calculator::BerekenBlok



Draft version 0.0.6
070712 – 02 Frontend Testversie 2
For .: Codename : Meera : .

Flowchart Verebus TOS > BerekenSeinbeelden



Draft version 0.0.6
070712 – 02 Frontend Testversie 2
For .: Codename : Meera : .

7 TOS XML Specificatie

Dit hoofdstuk is bedoeld om te laten zien welke XML bestanden gebruikt worden in TOS. En welke gegevens u kunt verwerken in XML. TOS maakt gebruik van de volgende 4 XML bestanden. De XML bestanden vormen de interface tussen de invoer (invoer unit of handmatige invoer) en de calculator unit.

StatischTrein.xml StatischZichtafstand.xml
Traject.xml Treinen.xml

De calculator unit gebruikt deze 4 bestanden om informatie uit te lezen die nodig is voor de berekening. De calculator schrijft niet naar XML bestanden.

TOS Frontend leest ook deze bestanden behalve StatischZichtafstand. TOSFrontend doet niets met StatischZichtafstand. Door het lezen van bijvoorbeeld StatischTrein.xml kan TOSFrontend de gebruiker een lijst met beschikbare treintypes aanbieden. TOSFrontend schrijft ook naar Traject.xml en Treinen.xml.

	TOS Calculator	TOS Frontend
StatischTrein.xml Node: <TreinStatisch>	alleen lezen	alleen lezen
StatischZichtafstand.xml Node: <ZichtafstandStatisch>	alleen lezen	- niet gebruikt -
Traject.xml Node: <Traject>	alleen lezen	lezen / schrijven
Treinen.xml Node: <TreinXML>	alleen lezen	lezen / schrijven

Tabel 7.1 overzicht van XML bestanden in TOS

Dit om de gebruiker te laten zien wat er al in XML staat. TOS om de Deze bestanden worden door de calculator unit gelezen om

7.1 XML Character encoding

Character encoding: UTF-8
 GEEN BYTE ORDER MARKS

Decimaal scheidingsteken: komma: ,

XML Header: <?xml version="1.0" standalone='yes' ?>

Indenting: 1 spatie, behalve voor XML Header en eerste daaropvolgende element <.

Case: Case sensitive

Remarks: Het gebruik van remarks is toegestaan.
 Echter deze worden door TOS Frontend verwijderd.

7.2 StatischTrein.xml

Doel van dit bestand

In StatischTrein.xml zijn de acceleratie en deceleratie constanten opgeslagen van ieder treintype. In hoofdnode TreinStatisch.

Inhoud van dit bestand

Dit bestand begint altijd met de XML header en de hoofdnode bestaande uit:

```
<?xml version="1.0" standalone='yes' ?>
<TreinStatisch Title="Statische parameters van Treinen">
```

Vervolgens voor ieder treintype:

```
<Type Title="DD-AR4">
```

Title is een string, verplicht aanwezig en ongelijk aan andere treintypes.

In node type:

```
<ad>-0,66</ad>
<a0>0,5</a0>
```

ad = deceleratie constante van de trein

a0 = acceleratie constante van de trein vanaf km/h

a1, a2... enz

ax is een double opgeslagen als tekst, **voor iedere mogelijke snelheid in gehele kilometers/h moet een acceleratie constante worden opgeslagen**. In andere woorden: als een trein 160 km/h kan rijden, dan zijn alle waarden a0 t/m a160 aanwezig.

De hoogste waarde voor a is meteen ook de maximale snelheid van de trein. De waarden moeten van laag naar hoog in XML worden opgeslagen.

Voorbeeld van (een deel van) StatischTrein.xml:

```
<?xml version="1.0" standalone='yes' ?>
<TreinStatisch Title="Statische parameters van Treinen">
  <Type Title="DD-AR4">
    <ad>-0,66</ad>
    <a0>0,5</a0>
    <a1>0,5</a1>
    ...
    <a160>0,5</a160>
  </type>
  <Type...
</Type>
</TreinStatisch>
```

7.3 StatischZichtafstand.xml

Doel van dit bestand

De minimale afstand in m voordat de machinist een sein ziet, bij een bepaalde snelheid is opgeslagen in StatischZichtafstand.xml. In hoofdnode ZichtafstandStatisch.

ZichtafstandStatisch geeft voor bepaalde snelheden (v) aan hoeveel meter (s) de zichtafstand is.

Inhoud van dit bestand

Dit bestand begint altijd met de XML header en de hoofdnode bestaande uit:

```
<?xml version="1.0" standalone='yes' ?>
<ZichtafstandStatisch Title="Zichtafstand van seinen">
```

Vervolgens kan voor iedere snelheid een zichtafstand worden gedefinieerd, maar dit *hoeft* niet voor elke mogelijke snelheid. We gebruiken een tag zonder titel:

```
<Zichtafstand>
```

In node Zichtafstand:

```
<v>100</v>
<s>250</s>
```

v = is de snelheid km/h voor deze waarde van zichtafstand

s = de afstand m voor deze waarde van zichtafstand

v, s = double opgeslagen als tekst, gehele waarden

De waarden van v moeten van laag naar hoog in XML worden opgeslagen.

Voorbeeld van (een deel van) StatischZichtafstand.xml:

```
<?xml version="1.0" standalone='yes' ?>
<ZichtafstandStatisch Title="Zichtafstand van seinen">
  <Zichtafstand>
    <v>80</v>
    <s>200</s>
  </Zichtafstand>
  <Zichtafstand>
    <v>100</v>
    <s>250</s>
  </Zichtafstand>
  <Zichtafstand>
    ...
  </Zichtafstand> ...
</ZichtafstandStatisch>
```

Van 0 tot 100 is de zichtafstand 200 m, van 100 tot v(3) is de zichtafstand 250 enz.

7.4 Traject.xml

Doel van dit bestand

In Traject zijn alle blokken opgeslagen die samen het door te rekenen traject vormen.
Hoofdnode: Traject.

Inhoud van dit bestand

Dit bestand begint altijd met de XML header en de hoofdnode bestaande uit:

```
<?xml version="1.0" encoding="UTF-8"?>  
<Traject Title="String" Remark="String">
```

Hierin zijn Title en Remark optionele strings waarin informatie over dit traject kan worden opgeslagen.

Vervolgens per blok:

```
<Blok Title="String">
```

Hierin is Title een optionele string waarin de naam van dit blok kan worden opgeslagen.

In node Blok:

```
  <Type>Hoofdsein</Type>  
  <Lengte>1000</Lengte>  
  <Bloksnelheid>100</Bloksnelheid>  
  <Bloksnelheid2>60</Bloksnelheid2>
```

Type = String, verplicht met mogelijke waarden: Hoofdsein, Voorsein, Halte of Snelheidsbord.

Lente = Lengte van het blok in meter, double opgeslagen als tekst, geheel getal

Bloksnelheid = Snelheid (groen of bloksnelheid) in km/h, double opgeslagen als tekst, geheel getal

Bloksnelheid 2 = Niet geïmplementeerd, maar wel in XML aanwezig, zie Bloksnelheid

En alleen in het eerste Blok:

```
  <Resolutie>1</Resolutie>
```

Resolutie = Resolutie van de berekening, kan zijn 1, 0,1 of 0,01 of experimenteel ongetest!: 0,001. Double opgeslagen als tekst.

De blokken moeten chronologisch in XML worden opgeslagen.

Voorbeeld van (een deel van) Traject.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<Traject Title="Test Traject voor Verebus TOS"
Remark="Variant: Plaats hier een opmerking over dit traject.">
  <Blok Title="14DT">
    <Type>Hoofdsein</Type>
    <Lengte>1000</Lengte>
    <Bloksnelheid>100</Bloksnelheid>
    <Bloksnelheid2>60</Bloksnelheid2>
    <Resolutie>1</Resolutie>
  </Blok>
  <Blok Title="15DT">
    <Type>Voorsein</Type>
    <Lengte>1000</Lengte>
    <Bloksnelheid>100</Bloksnelheid>
    <Bloksnelheid2>60</Bloksnelheid2>
  </Blok>
  <Blok Title="17DT">
    <Type>Hoofdsein</Type>
    <Lengte>1000</Lengte>
    <Bloksnelheid>160</Bloksnelheid>
    <Bloksnelheid2>60</Bloksnelheid2>
  </Blok>
  <Blok ... </Blok>
</Traject>
```

7.5 Treinen.xml

Doel van dit bestand

In treinen zijn de trein parameters opgeslagen van trein 1 en trein 2, die worden gebruikt om door het traject te rijden. HoofdnodetreinXML.

Inhoud van dit bestand

Dit bestand begint altijd met de XML header en de hoofdnodetreinXML bestaande uit:

```
<?xml version="1.0" encoding="UTF-8"?>
<treinXML Title="Parameters van Treinen (Gegenereerd)">
```

Vervolgens voor iedere trein:

```
<Trein Title="Trein 1: IRM3">
```

Hierin is Title een optionele string met een trein titel

In node Trein:

```
<Type>IRM3</Type>
<Lengte>100</Lengte>
<Beginsnelheid>10</Beginsnelheid>
```

Type = een type dat bekend is in TreinStatisch

Lengte = is treinlengte in m, double opgeslagen als tekst, geheel getal

Beginsnelheid = de snelheid in km/h bij het binnenrijden van het traject.

double opgeslagen als tekst, geheel getal <=
grootste a waarde bekend in TreinStatisch

De treinen moeten chronologisch in XML worden opgeslagen.

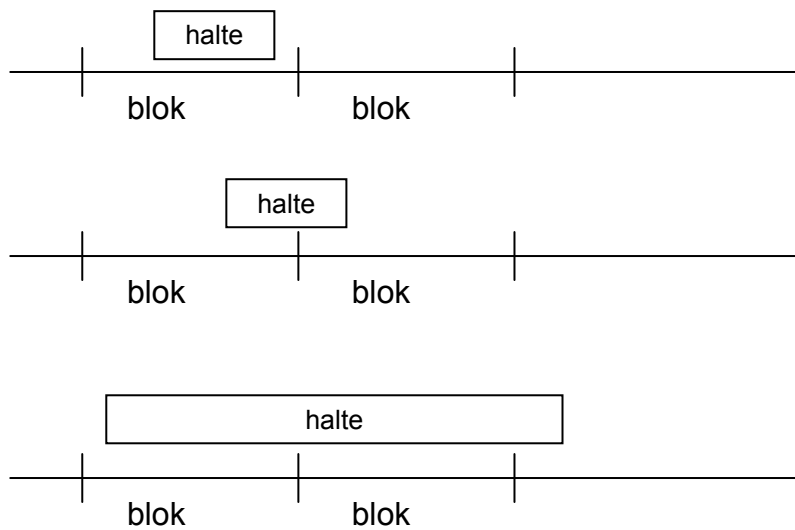
Voorbeeld Treinen.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<treinXML Title="Parameters van Treinen (Gegenereerd)">
  <Trein Title="Trein 1: IRM3">
    <Type>IRM3</Type>
    <Lengte>100</Lengte>
    <Beginsnelheid>10</Beginsnelheid>
  </Trein>
  <Trein Title="Trein 2: IRM3">
    <Type>IRM3</Type>
    <Lengte>100</Lengte>
    <Beginsnelheid>10</Beginsnelheid>
  </Trein>
</treinXML>
```

8 Advies voor implementatie van haltes

In deze paragraaf vind u een kort advies voor de implementatie van haltes. Dit is met name bedoeld voor mijn opvolger, of de programmeurs in het vervolgproject. Mijn advies zal zich beperken tot de implementatie van haltes. Aangezien dit de belangrijkste vraag van Verebus zal zijn in het vervolgproject.

In de eerste plaats is het belangrijk om te realiseren dat een halte NIET is gerelateerd aan een blok. De onderstaande figuur kan hierover meer duidelijkheid bieden:



Figuur 8.1 Haltes en blokken

De verticale streep is ter aanduiding waar de las zit. Door middel van de las kan worden gedetecteerd of een trein zich in een bepaald blok bevindt. Het is het meest logisch om een las ná een halte te plaatsen. Maar dit *hoeft* niet.

Doordat haltes niet aan een blok zijn gerelateerd, moeten deze op een andere manier worden geïmplementeerd dan de bestaande elementen in TOS. De bestaande elementen hoofdein, vorsein en bloksnelheid zijn *wel* blok gerelateerd.

Implementatie van haltes kan het beste plaats vinden door een nieuw node type toe te voegen aan Traject.xml. Bijvoorbeeld zoals hier:

```
<Halte Title="Station Voorhout" >
  <Type>Halte</Type>
  <s>1123</s>
  <t>180</t>
</Halte>
```

Waarin s de afstand is waarop zich de halte bevind in meters en t de wachttijd is seconden.

- We laten nu de trein door het traject rijden, tot deze volgens de berekening op punt s is aangekomen.
- Vervolgens gebruiken we de vectoren RTGlobalTijdVector, RTGlobalAfgelegdVector uit de klasse Calculator om te berekenen wanneer de remming moet worden ingezet.
- Vanaf dat punt overschrijven we de vector, en laten we de trein vertragen naar 0 km/h.
- Hierna tellen we de wachttijd op bij RTGlobalTijdVector
- Hierna vervolgen we de berekening met Calculator::BerekenBlok op de voor TOS gebruikelijke manier.

Door de node halte uit te breiden met een extra node waarin staat *watvoor* treinen stoppen bij een halte. Bijvoorbeeld stoptrein, sneltrein of intercity. Kan een uitgebreidere implementatie worden gemaakt voor de haltes. Hierdoor kan het mogelijk worden trein 1 te laten stoppen op halte a en niet op halte b. En trein 2 te laten stoppen op halte b en niet op a.

Deze halte types zouden dan ook aan TreinXML moeten worden toegevoegd.

Slotwoord

Ik hoop dat dit document u inzicht heeft gegeven in de werking van het Verebus Trein Opvolgtijden Systeem (TOS). En dat de informatie in dit document voldoende zal zijn om TOS in de huidige vorm te gebruiken en in de toekomst verder te ontwikkelen. Het verder ontwikkelen van TOS is als uitgangspunt gebruikt tijdens het schrijven van deze documentatie.

Graag wil ik nog de volgende collega's bedanken. Zij hebben mij geholpen tijdens dit project met problemen waar ik zelf zo snel de oplossing niet van zag:

Bert (Meh) Fraterman	C++ gerelateerde onderwerpen "Hee Bert is dat fotoboek nu al af?"
Dimitri (DimVim) Princen	C++ gerelateerde onderwerpen Veel succes met afstuderen Dim! En veel plezier in je nieuwe baan!

En de volgende persoon voor het leveren van beeldmateriaal voor in de documentatie en op mijn site. En zijn toestemming om dit materiaal te gebruiken:
Max Hovens Beeldmateriaal en achtergronden over seinen

En verder Jelle de Jong 'mental support' en Meera Nathwani (coaching).

Nogmaals wens ik u veel succes toe bij het verder ontwikkelen en gebruiken van Verebus TOS. Voor vragen of informatie over vervolgprojecten kunt u zich wenden tot Dhr. W.J.T. de Kaper (W.J.T.deKaper@hhs.nl).

Barry de Graaff

Bijlagen

- 001 Startdocument Verebus TOS definitief 0.0.4b.pdf
- 002 Voorstel voor aflevering Verebus TOS.pdf