Programmeerproject 2: Voorstudie (fase 1)

Maxim Brabants

0576581

Maxim.Lino.Brabants@vub.be

Academiejaar 2021-2022

Inhoudsopgave

1 Korte inleiding

Voor het tweede programmeerproject binnen het academiejaar 2021-2022 wordt er van ons verwacht dat we een volledig controlesysteem gaan uitbouwen dat zal instaan voor de aansturing van een modelspoor inclusief de treinen die daarover zullen kunnen rijden. Dit alles zal volledig programmatorisch moeten gebeuren met een opsplitsing in enkele onderdelen. Eerst een kort overzicht van welke onderdelen precies aanwezig moeten zijn en welke aspecten per onderdeel van belang zijn (op deze manier denk ik na over wat precies de bevoegdheid is van elk onderdeel en wat de mogelijkheden zijn):

- Allereerst moet er iets aanwezig zijn dat rechtstreeks met de spoorelementen kan communiceren zodat er op bepaalde momenten een verandering in de spoorsituatie kan plaatsvinden. Hiervoor zal het **Command Station (Z21)** verantwoordelijk zijn. Deze zal a.d.h.v. een bepaald protocol met de hardware communiceren (deze software is reeds voorzien). Dit station gaan we kunnen aansturen door er van buitenaf commando's naar te sturen.

- Vervolgens hebben we nog een permanent draaiend component dat zal communiceren met de modelbouw hardware. Die software zal voortdurend moeten actief blijven omdat deze juist de signalen zal sturen naar het Command Station. Dit component kunnen we laten draaien op onze computer of afzonderen op een raspberry Pi. **(Infrabel)**

- Als allerlaatst hebben we nog een component met een grafische interface die eigenlijk los moet staan van de logica voor het aansturen van de sporen, maar wel moet instaan voor het berekenen van bijvoorbeeld trajecten. **(NMBS)**

De grote nadruk zal ongetwijfeld ook liggen op de assemblage van al deze elementen tot één groot geheel. We mogen namelijk geen elementen gaan mixen met elkaar, want als we dat gaan doen, wordt het zowel voor ons als voor andere developers onduidelijk waar we elk element kunnen terugvinden.

2 Beschrijving ADTs

2.1 Locomotief

Locomotieven vormen een integraal onderdeel binnen dit project. Het algemene idee zegt ons dat we ten alle tijden moeten trachten de garantie te bewaren dat een trein veilig van punt A naar punt B kan reizen. M.a.w. dat een trein zonder enige obstakels (botsingen, ontsporingen,…) bepaalde trajecten moet kunnen afleggen. Qua gedrag denk ik aan het kunnen starten of stoppen van een trein, alsook het versnellen en vertragen van een rijdende locomotief en ten slotte ook de bepaling van de huidige rijrichting (vooruit of achteruit).

Operaties	**Signatuur**
start!	(ϕ → ϕ)
stop!	(ϕ → ϕ)
versnel!	(ϕ → ϕ)

vertraag!	(ϕ → ϕ)
veranderRijrichting!	(ϕ → ϕ)

- **start!** zal het signaal geven dat de locomotief in beweging is naar de huidige richting toe.
- **stop!** zal dan weer aangeven dat de trein niet meer rijdt.

2.2 Wissel

Een wissel op zich voorziet niet echt veel functionaliteit. Het voornaamste dat deze moet kunnen doen is de richting van het spoor bepalen, dat vermoedelijk zal gebeuren door ergens in de Z21-bibliotheek een parameterloze procedure aan te roepen die vervolgens naar die bepaalde wissel het commando zal sturen om de stand te veranderen. Voor later moeten we steeds in ons achterhoofd houden dat er verschillende wisselopstellingen bestaan en dat we deze zo goed mogelijk op elkaar laten inwerken.

Operaties	**Signatuur**
toggle-wissel!	(ϕ → ϕ)
geefStatus	(ϕ → number)

- De **toggle-wissel!** gaat de effectieve wissel doorvoeren en zorgt dat er een fysieke verandering plaatsvindt op het spoor. De stand van de wissel wordt hierbij veranderd.
- We zouden een **geefStatus** operatie kunnen voorzien die de huidige stand van de wissel kan teruggeven (1 of 2), maar dit is niet noodzakelijk, want de stand van de wissel zal waarschijnlijk een attribuut zijn van het Wissel-ADT dus deze kunnen we rechtstreeks teruggeven.

2.3 Detectieblok

Er moeten ook elementen in de spooropstelling aanwezig zijn die ons op een gegeven moment informatie gaan geven over het mogelijke voorkomen van een bepaalde trein in een bepaalde zone. Zo kan ons programma anticiperen en eventueel bepaalde wissels doorvoeren. We houden hierbij rekening dat we binnen zulke detectieblokken slechts weten of een trein daar aanwezig is of niet en niet de exacte locatie van een trein binnen een detectieblok.

Operaties	**Signatuur**
bezet?	(ϕ → Boolean)
geefNummer	(ϕ → number)

- **bezet?** kan ons voor een bepaald detectieblok zeggen of er een locomotief binnen dat blok aan het rijden is.

- Net zoals bij de wissel kunnen we hier aan de hand van een procedure het nummer/zone van een bepaald detectieblok teruggeven of rechtstreeks als attribuut opnemen.

2.4 Infrabel

Dit zie ik als het centraal ADT dat voortdurend blijft opereren en waar alle logica rond het veilig rijden van de treinen, togglen van wissels en registreren van detectieblokken in verzameld staat. Dit ADT zal dan ook verantwoordelijk gesteld zijn om op de juiste momenten wissels uit te voeren en signalen van detectieblokken te versturen zodat het welzijn van de treinen bewaard blijft. Hierin gaan alle wisselcomponenten, locomotieven en detectieblokken geregistreerd staan. In het geval dat er bijvoorbeeld werken aan het spoor zijn, is het de verantwoordelijkheid van Infrabel om hier een oplossing voor te bieden. Analoog bij een eventuele verandering in de sporeninfrastructuur, moet Infrabel ervoor zorgen dat treinen hun trajecten kunnen blijven afrijden en dat er rekening wordt gehouden met de veranderde spoorsituatie.

Operaties	**Signatuur**
status-detectieblokken	(ϕ → assoc-List <number, Boolean>)
status-detectieblok	(number → Boolean)
voer-wissel-uit!	(number → ϕ)
pas-snelheid-aan!	(Locomotief → ϕ)
loop!	(ϕ → ϕ)
botsing?	(Locomotief, Locomotief → ϕ)

- Bij Infrabel houden we al onze detectieblokken bij. Door over al deze blokken te lopen kunnen we hun statussen verkrijgen en op basis daarvan actie ondernemen.

- We zouden dit ook kunnen opvragen voor een enkel detectieblok.

- Infrabel moet wissels kunnen doorvoeren op het spoor aan de hand van een gegeven wissel/wisselnummer.

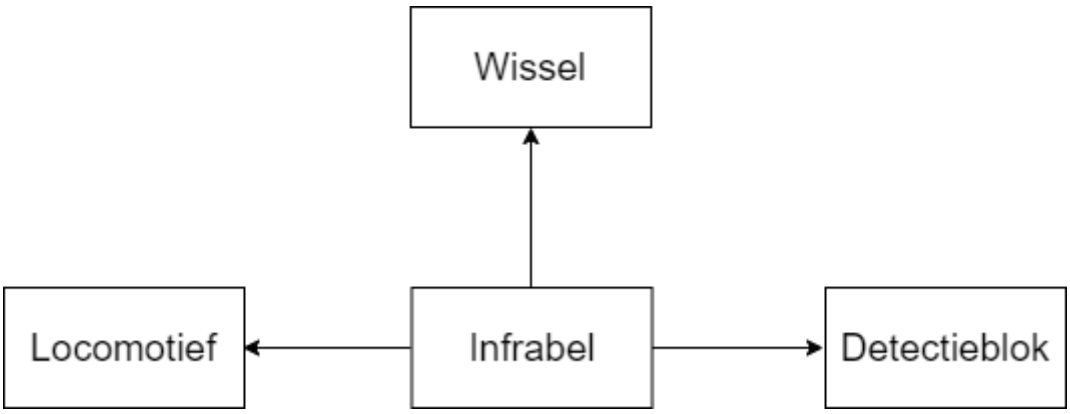
- De snelheid van een locomotief zou eventueel kunnen aangepast worden.

- Er moet een procedure zijn die non-stop blijft uitvoeren en die dan constant de logica blijft berekenen. (zoals de update-callback! In pp1)

- Eventueel een procedure die nakijkt of twee treinen in aanmerking komen om mogelijks te botsen.

3 Afhankelijkheidsdiagram

Er zijn bepaalde types die niet correct gaan kunnen werken zonder het bestaan van andere. Er is dus een bepaalde afhankelijkheid die dient gerespecteerd te worden om een goede werking van het gehele programma te kunnen garanderen.



Dit is een overzicht dat de basis weergeeft van het systeem. Er zullen hoogstwaarschijnlijk nog meerdere ADT's bijkomen in de loop der tijd waar ik op dit moment niet kan aan denken, maar dit diagram zegt eigenlijk dat Infrabel alle componenten samenvoegt en op basis daarvan de logica gaat berekenen (wissels doen indien nodig, snelheden aanpassen aan de situatie).

4 Werktermijn

Een overzicht van de verwachte taken per week :

Week	**Taak**
Fase 1	
Week 5	Voorstudie schrijven en afwerken.
Week 6	Experimenteren in de simulator (wissels uitlezen en verzetten, locomotieven laten rijden, detectieblokken uitlezen,…)
Week 7	Starten met maken van GUI. (Racket Graphical Interface Toolkit)
Week 8	Spoorinformatie tonen in GUI.
Week 9	Interactie voor de gebruiker toevoegen.
Week 10	De GUI op punt krijgen + verslag schrijven.
Week 11	Indienen code en documenten fase 1.