VRIJE
UNIVERSITEIT
BRUSSEL

PROGRAMMEERPROJECT 2

Documentatie fase 3

Gérard Lichterttt
gerard.Lichtert@vub.be
0557513

11 mei 2021

Sciences and Bio-Engineering Sciences

Inhoudsopgave

1 Inleiding

Dit document bevat het verslag van fase 3 van het vak "Programmeerproject 2". Het behandelt de eerste functionele vereisten en de 3 uigebreide vereisten. Vervolgens zal het over de gebruikte datastructuren gaan en het afhankelijkheids diagram. Deze datastructuren zullen er voor zorgen dat alle functionaliteiten werken en vanuit de GUI aangeroepen kunnen worden. Verder staat in dit verslag een beschrijving van de API tussen de infrabel- en NMBS-component, de planning en het logboek.

2 Functionele vereisten

In dit verslag worden de functionele vereisten per fase besproken. In de eerste fase werd de GUI gemaakt alsook de command and control. De code van de command en control moeten we zelf niet schrijven maar de code die de hardware aanstuurt wel. De code moet locomotieven kunnen laten starten en stoppen, hun snelheid en rijrichting aflezen en veranderen. Verder moeten we ook de de stand van de wissels kunnen uitlezen en verzetten. Via de detectieblokkken detecteren we waar een trein zich bevindt.
De GUI laat de eindgebruiker de toestand van het spoornetwerk en de locomotieven zien. Het laat ook interactie toe met de wissels en locomotieven.
In de tweede fase was het de bedoeling dat we de eerste twee uitgebreide vereisten implementeren. Voor dit project werd gekozen voor botsingpreventie aan de hand van een reservatie en bezettingsysteem. Dit zal ervoor zorgen dat een trein zijn snelheid op nul gezet wordt als het pad dat die gaat afleggen al gereserveerd of bezet is. Aangezien de stand van de wissels op eender welk moment gewijzigd kan worden, worden er veel spoorsegmenten gereserveerd tot de aanliggende detectieblokken. Dit is wel met de voorwaarde dat ze op het pad zijn van de richting van de trein. Een andere uitgebreide vereiste waar voor gekozen werd is het automatisch trajectbeheer. De manier waarop dit geimplementeerd is dat de eindbestemming ingevoerd wordt en het pad berekend word. Dit wordt slechts gedaan als de trein detecteerbaar is. Anders zal het pad nog eens herberekend worden wanneer die detecteerbaar is. Er wordt bij het trajectbeheer ook rekening gehouden met het reservatiesysteem om botsingen te voorkomen.
In de 3e fase is het de bedoeling dat we de laatste uitgebreide vereiste implementeren. Bij dit project is dit de Raspberry Pi vereiste. Dit houdt in dat het mogelijk moet zijn om infrabel te runnen op de Raspberry Pi en NMBS op de computer. Dit wordt behaald door het externe IP adres (of lokaal) mee te geven bij het aanmaken van het NMBS object.

3 ADT's

3.1 Track ADT

Het track ADT houdt bij welke spoorsegmenten verbonden zijn. Het houdt ook bij of het spoorsegment al dan niet gereserveerd is. De reservatiestand kan ook aangepast worden.

Naam	Signatuur	Beschrijving
new	(symbol, list → track%)	Maakt een track object aan. Verwacht de ID van het spoorsegment en een lijst met de ID's van de verbonden spoorsegmenten.
get-track-id	(/ → symbol)	Geeft de ID van het spoorsegment terug.
track-links	(/ → vector)	Geeft de vector terug met de ID's van de verbonden spoorsegmenten.
set-links!	(list → /)	Verandert de verbonden spoorsegmenten.
links-map	((symbol → any) → vector)	Voert een procedure uit op de ID's van de verbonden spoorsegmenten en geeft de opgespannen vector terug.
reserved?	(/ → symbol ∪ false)	Geeft de ID van de locomotief die het spoorsegment gereserveerd heeft of false.
reserve!	(symbol → /)	Verandert de reservatiestatus naar het meegegeven ID.
cancel-reservation!	(/ → /)	Zet het reservatiestatus op false
nr-of-links	(/ → integer)	Geeft het aantal verbonden spoorsegmenten terug

Tabel 1: Signaturen van track%

3.1.1 Toelichting

Het track ADT is eigenlijk een superclasse van de volgende ADT's. Dit is zodat de volgende ADT's de methoden erven van het track ADT. New maakt een nieuwe track ADT aan. Dit kan ook vervangen worden door (make-object track% < argumenten >). $get-track-id$ dient om de ID op te vragen. $track-links$ dient om de vector met verbonden spoorsegmenten op te vragen. $set-links!$ wijzigt de verbonden spoorsegmenten. $links-map$ voert een procedure uit op de vector-elementen. $reserved?$ geeft of de ID van het locomotief terug die het spoorsegment gereserveerd heeft of false. $cancel-reservation!$ zet de reservatiestand naar false. $nr-of-links$ geeft het aantal verbindingen terug. Classen zoals switch en detectieblokken zullen de methoden erven van deze classe.

3.2 Switch ADT

Het switch ADT erft de methoden van het Track ADT. Het zal dus bovenop de onderstaande methoden ook de methoden van het Track ADT bevatten. Het switch ADT houdt de stand van een wissel bij, welk spoorsegment verbonden is met de huidige stand en het staat toe om de stand te veranderen.

Naam	Signatuur	Beschrijving
new	(symbol, list → switch%)	Maakt een nieuwe switch object aan
get-status	(/ → integer)	Geeft de stand van de wissel terug.
set-status!	(integer → integer)	Veranert de stand van de wissel.
get-merge-track	(/ → symbol)	Geeft de ID van het spoorsegment die niet met de stand van de wissels verandert.
get-linked-track	(/ → symbol)	Geeft de ID van het spoorsegment die verbonden is met de stand van de wissel.

Tabel 2: Signaturen van switch%

3.2.1 Toelichting

Het switch ADT is een subclasse van het track ADT. Het kan dus dezelfde methoden gebruik als het track ADT. De extra methoden zijn *new* dat dezelfde argumenten verwacht als het track ADT (De ID van het spoorsegment en een lijst met de ID's van de verbonden spoorsegmenten.) Let op!: Zet de ID op de juste index in de lijst. Als de stand van een wissel 1 is en het verbindt met het spoorsegment 1-5, dan zou de lijst '(U-1 1-5 ... etc...) moeten zijn. Dit is zodat de interne index van de stand van de wissels analoog is aan de stand van de wissels van de hardware. $get-status$ geeft de stand van de wissel terug als een nummer. $set-status!$ verandert de stand van de wissel naar het gegeven stand. Zorg er wel voor dat de index correct overeenkomt omdat het afleest van de vector die de verbonden spoorsegmenten bijhoudt. $get-merge-track$ geeft de ID van het spoorsegment die niet met de stand van de wissel verandert. $get-linked-track$ geeft de ID terug van de huidige verbonden spoorsegment aan de hand van de stand van de wissel.

3.3 Dblock ADT

Het detection block ADT is een subclasse van het track ADT. Het erft dus net zoals het switch ADT de methoden van het track ADT. Bovendien houdt het ook bij of dat er zich een locomotief op bevindt.

Naam	Signatuur	Beschrijving
new	(symbol, list → dblock%)	Maakt een nieuw detectie block object aan.
occupied?	(/ → symbol ∪ false)	Geeft ID van de locomotief terug die momenteel de detectieblok bezet.
occupy!	(symbol → /)	Bezet de detectieblok met het gegeven ID.
vacant!	(/ → /)	Maakt de detectieblok vrij.

Tabel 3: Signaturen van dblock%

3.3.1 Toelichting

De methoden die bruikbaar zijn voor het detectieblok ADT zijn de methoden die het ADT erft van het track ADT. Dit is ook niet zonder de hierboven genoemde methoden die het arsenaal vervullen. *new* maakt een nieuwe detectieblok object aan. Het verwacht dezelfde argumenten

zoals *new* van het track ADT. *occupied*? geeft de ID van de locomotief die het detectieblok bezet of false indien die niet bezet is. *occupy*! bezet het detectieblok met het gegeven ID van een locomotief. *vacant*! zal er voor zorgen dat het detectieblok terug vrij is.

3.4 Locomotive ADT

Het locomotief ADT houdt de interne data van een locomotief bij zoals snelheid, richting, vorige locatie, huidige locatie, manuele modus, het pad en de eindbestemming. Dit komt natuurlijk ook met methoden om ze te veranderen. Het pad en eindbestemming wordt gebruikt voor het automatisch trajectbeheer.

Naam	Signatuur	Beschrijving
new	(symbol, symbol, symbol, symbol → locomotive%)	Maakt een nieuw lovomotief object aan.
reserve!	(list → /)	Slaat een lijst van ID's van spoorsegmenten op die de locomotief gereserveerd heeft.
clear-reservations!	(/ → /)	Verwijdert de lijst van reservaties.
made-reservations?	(/ → list ∪ false)	Geeft een lijst van ID's van de gereserveerde spoorsegmenten terug.
manual-override	(boolean → /)	Wijzigt de reservatieprotocol van de locomotief.
manual?	(/ → boolean)	Geeft het reservatieprotocol terug van de locomotief.
get-loco-id	(/ → symbol)	Geeft de ID van de locomotief terug.

Tabel 4: Signaturen van locomotive%

3.4.1 Toelichting

new maakt een nieuw locomotief object aan. Het neemt als eerste de ID van de locomotief, de richting van de locomotief, de huidige locatie en de vorige locatie. $reserve$! verwacht een lijst van ID's van spoorsegmenten die de locomotief gereserveerd heeft. $clear-reservations$! verwijdert de lijst van ID's van de gereserveerde spoorsegmenten. De waarde hiervan verandert naar $false$. $made-reservations$? geeft of de lijst terug van de van de ID's van de gereserveerde spoorsegmenten of $false$ indien de locomotief geen spoorsegmenten gereserveerd heeft. $manual-override$ zal dienen om de status van de manier van het reservatiesysteem te veranderen. Wanneer de status *true* is zal het de bezetting van een detectieblok negeren, dit zal vooral dienen om naar dichtbijzijnde detectieblokken te kunnen navigeren zonder rekening te houden met locomotieven die te dicht in de buurt zijn. De enige argumenten die dus gebruikt kunnen worden hiervoor zijn dus ook de booleans. *manual*? geeft de status van de manuele modus terug. Als laatste $get-loco-id$ geeft de ID terug van de locomotief.

3.5 Railway ADT

Het railway ADT brengt de de spoor gerelateerde ADT's zoals het track ADT, switch ADT, het detectieblok ADT en het locomotief ADT samen. Zo houdt he railway ADT het volledig spoornetwerk samen. Het netwerk zelf wordt bewaard in een graaf. Het programma maakt hierdoor gebruik van de *graph* library. Buiten een graaf van het spoornetwerk zelf bewaart het ook een

graaf met detectieblokken met bogen naar die gedefinieerd zijn op vlak van bereikbaarheid. Er is bijvoorbeeld dus een boog tussen detectieblok 1-4 en 1-1 omdat er een pad is van deze detectieblokken die waarbij er geen andere detectieblok tussen zit en dat de trein niet van richting moet veranderen tussen deze detectieblokken (wel op de detectieblok zelf!). Op basis hiervan kunnen we dus trajecten berekenen. Hiervan en een speciale graaf die alleen de detectieblokken bevat. Dit is zodat we kunnen navigeren en paden kunnen berekenen aan de hand van paden tussen detectieblokken. De methoden van het Railway ADT zullen voornamelijk zoek-functionaliteiten aanbieden. Een voorbeeld hiervan is als we een locomotief object willen aanspreken dan zal het dit eerst opzoeken in de hashmap waar het opgeslagen staat. Hetzelfde geldt voor het aanspreken van track ADT en zijn subclassen.

3.5.1 Toelichting

3.6 Infrabel ADT

3.6.1 Toelichting

De methoden die bruikbaar zijn voor het detectieblok ADT zijn de methoden die het ADT erft van het track ADT. Dit is ook niet zonder de hierboven genoemde methoden die het arsenaal vervullen. *new* maakt een nieuwe detectieblok object aan. Het verwacht dezelfde argumenten zoals *new* van het track ADT. *occupied*? geeft de ID van de locomotief die het detectieblok bezet of false indien die niet bezet is. *occupy*! bezet het detectieblok met het gegeven ID van een locomotief. *vacant*! zal er voor zorgen dat het detectieblok terug vrij is.

3.7 Locomotive ADT

Het locomotief ADT houdt de interne data van een locomotief bij zoals snelheid, richting, vorige locatie, huidige locatie, manuele modus, het pad en de eindbestemming. Dit komt natuurlijk ook met methoden om ze te veranderen. Het pad en eindbestemming wordt gebruikt voor het automatisch trajectbeheer.

3.7.1 Toelichting

3.8 Gui ADT

3.8.1 Toelichting

3.9 NMBS ADT

3.9.1 Toelichting

4 Afhankelijkheids diagram

5 Beschrijving van API tussen infrabel- en NMBS-component

6 Planning

7 Logboek

Naam	Signatuur	Beschrijving
new	(symbol, symbol, symbol, symbol → locomotive%)	Maakt een nieuw lovomotief object aan.
get-list-of-tracks	(/ → list)	Geeft een lijst van terug van de hashmap van alle spoorsegmenten.
get-track	(/ → track ∪ dblock ∪ switch)	Geeft het gezochte spoor object terug.
make-track	(symbol, list → track)	Maakt een track object aan.
make-switch	(symbol, list → switch)	Maakt een wissel object aan.
make-dblock	(symbol, list → dblock)	Maakt een detectieblok object aan.
add-track!	(track ∪ dblock ∪ switch → track ∪ dblock ∪ switch)	Voegt het spoor object toe aan het spoornetwerk.
remove-track!	(symbol → /)	Verwijdert het gezochte object uit het spoornetwerk.
update-track!	(symbol → (track ∪ dblock ∪ switch → /))	Voert de meegegeven procedure uit op het gezochte object.
for-each-track	((track ∪ dblock ∪ switch → any) → any)	Voert een procedure uit op alle spoorsegmenten
for-each-link	((track ∪ dblock ∪ switch → any), symbol → vector)	Voert een procedure uit op de aanliggende spoorsegmenten van het gezochte spoorsegment.
get-railway-graph	(/ → graph)	Geeft de graaf met het spoornetwerk terug.
get-dblock-graph	(/ → graph)	Geeft de graaf met alleen detectieblokken terug.
make-locomotive	(symbol, symbol, symbol, symbol → locomotive)	Maakt een locomotief object aan aan de hand van de gegeven ID, richting, huidige detectieblok en vorige detectieblok.
add-locomotive!	(locomotive → /)	Voegt een locomotief object toe aan het spoornetwerk.
remove-locomotive!	(symbol → /)	Verwijdert het gezochte locomotief object van het spoornetwerk.
update-locomotive!	(locomotive → /)	Verandert gegevens van een locomotief door het te vervangen met een nieuwe.
get-locomotive	(symbol → locomotive ∪ false)	Geeft het gezochte locomotief object terug of false indien deze niet bestaat.
for-each-loco	((locomotive → any) → any)	Voert een procedure uit op alle locomotief objecten.

Tabel 5: Signaturen van railway%

Naam	Signatuur	Beschrijving
new	(symbol, list → dblock%)	Maakt een nieuw detectie block object aan.
occupied?	(/ → symbol ∪ false)	Geeft ID van de locomotief terug die momenteel de detectieblok bezet.
occupy!	(symbol → /)	Bezet de detectieblok met het gegeven ID.
vacant!	(/ → /)	Maakt de detectiblok vrij.

Tabel 6: Signaturen van dblock%

Naam	Signatuur	Beschrijving
new	(symbol, symbol, symbol, symbol → locomotive%)	Maakt een nieuw lovomotief object aan.

Tabel 7: Signaturen van Infrabel%