Programmeerproject 2: Specificatie “Railway system” Fase II

Rayane Kouidane

2de Bachelor Computerwetenschappen [Rayane.Kouidane@vub.be](mailto:Rayane.Kouidane@vub.be) N°0587073

Academiejaar 2022-2023

Inhoudsopgave

1. [Introductie 2](#_bookmark0)
2. [Algemene beschrijving 2](#_bookmark1)
3. [Software-architectuur 2](#_bookmark2)
4. [ADT’s 3](#_bookmark3)
   1. [Train ADT 3](#_bookmark4)
   2. [Detection-block ADT 3](#_bookmark5)
   3. [Switch ADT 3](#_bookmark6)
   4. [Gui ADT 3](#_bookmark7)
   5. [Railway ADT 4](#_bookmark8)
   6. [Infrabel ADT 5](#_bookmark10)
   7. [NMBS ADT 5](#_bookmark10)
   8. [Andere 6](#_bookmark11)
5. [Gebruikte bronnen 6](#_bookmark12)

# Introductie

Dit bestand bevat de specificatie van de tweede fase van het project. Eerst zal ik een algemene beschrijving geven van wat de vereisten zijn voor de tweede fase. Daarna een beschrijving geven van de software-architectuur met een afhankelijkheidsdiagram. Als laatst zal ik de verschillende componenten bespreken.

# Algemene beschrijving

Voor de tweede fase van het project moesten we twee vereisten kiezen uit een gegeven lijst en deze implementeren. Als vereiste heb ik gekozen voor *Tijdstabellen uitlezen uit een bestand* en *Dynamische snelheden*. Omdat ik dynamische snelheden heb gekozen, zal ik in de derde fase moeten kiezen voor *Automatische trajectberekening* als derde vereiste.

# Software-architectuur

NMBS

Infrabel

Help-functions

GUI

Railway ADT

Interface (Simulator)

Train ADT

Switch ADT

Detection- block ADT

Racket/GUI

Racket/Date

Main

Trajectories/

*Figuur 1: Afhankelijkheidsdiagram Fase II*

* Main is afhankelijk van NMBS en Infrabel, omdat we daarin het controlesysteem kunnen opstarten door gebruik te maken van de procedures uit NMBS en Infrabel.
* Infrabel is afhankelijk van het NMBS ADT, Railway ADT, Interface (Simulator) en Help- functions. Help-functions is een bestand dat hulpprocedures bevat. Dit bestand is zelf afhankelijk van Racket/Date. Via procedures uit de interface, het NMBS ADT en Railway ADT kan Infrabel het spoornetwerk van verschillende onderdelen wijzigen. Deze zijn: de simulator, nmbs en objecten. Daarom is het Railway ADT afhankelijk van het Train, Switch en Detection-block ADT.
* NMBS is afhankelijk van Infrabel, Railway ADT en de GUI. Het zal procedures uit Infrabel gebruiken en deze koppelen aan de GUI. Zodat wanneer de gebruiker bijvoorbeeld de snelheid van een trein verandert, dan gebeurt dit zowel in de simulator als in de railways (railway-infrabel en railway-nmbs). De GUI gebruikt hulpprocedures uit Help-functions en ook Racket/GUI. Hiermee kan het knoppen, sliders… tekenen.
* Wanneer een gebruiker een traject wilt starten, dan zal NMBS het juiste traject moeten opzoeken in het bestand Trajectories en deze vertalen naar de juiste instructies.
* Voor het Train ADT, Switch ADT, Detection-block ADT, Railway ADT en de dynamische snelheid zijn er unit-tests. Voor de andere zijn er geen tests omdat het moeilijk is om te controleren of iets wel of niet werkt via RackUnit. Je kan dit enkel controleren door het geheel te starten.
* (Voorlopig gebruikt Infrabel procedures uit NMBS en NMBS procedures uit Infrabel, later zullen ze, via TCP, met elkaar communiceren)

# ADT’s

## Train ADT

|  |  |
| --- | --- |
| Naam | Signatuur |
| **make-train** | symbol → Train |
| **id** | ∅ → symbol |
| **speed** | ∅ → number |
| **speed!** | number → ∅ |
| **direction** | ∅ → boolean |
| **inverse-direction!** | ∅ → ∅ |
| **prev-dtb** | ∅ → symbol |
| **prev-dtb!** | symbol → ∅ |
| **trajectory?** | ∅ → boolean |
| **trajectory!** | ∅ → ∅ |

*Tabel 1: Operaties Train ADT*

Het Train ADT bevat alle procedures die nodig zijn om informatie te vragen over de trein: de id van de trein (id), de snelheid (speed), de richting (direction), de vorige positie (prev-dtb) en een boolean (trajectory?) die #t is wanneer een trein een traject volgt. Het bevat ook procedures om die informatie te wijzigen: speed!, inverse-direction! en prev-dtb! wijzigen de snelheid, richting en vorige positie van de trein. trajectory! verandert de waarde van trajectory? in het tegengestelde van de huidige waarde.

## Detection-block ADT

|  |  |
| --- | --- |
| Naam | Signatuur |
| **make-detection-block** | symbol → Detection-block |
| **id** | ∅ → symbol |
| **used?** | ∅ → boolean |
| **change-status!** | ∅ → ∅ |
| **last-train** | ∅ → symbol |
| **last-train!** | symbol → ∅ |

*Tabel 2: Operaties Detection-block ADT*

Het Detection-block ADT representeert een detectie-block. Deze bevat de volgende informatie: id van de detectieblok, is een detectieblok wel of niet bezet en de laatste trein die op de detectieblok was.

Daarnaast kan je met change-status! en last-train! deze informatie wijzigen.

## Switch ADT

|  |  |
| --- | --- |
| Naam | Signatuur |
| **make-switch** | symbol → Switch |
| **id** | ∅ → symbol |
| **state** | ∅ → number |
| **state!** | number → ∅ |

*Tabel 3: Operaties Switch ADT*

Het Switch ADT bevat de procedures om informatie te vragen over een wissel en de staat ervan te wijzigen. De procedure id en state geven respectievelijk de id van de switch en de staat terug. state! wijzigt de positie van de wissel.

## Gui ADT

|  |  |
| --- | --- |
| Naam | Signatuur |
| **make-gui** | ∅ → Gui |
| **start** | (∅ → ∅) → ∅ |
| **exit** | (∅ → ∅) → ∅ |
| **draw-train** | (symbol → ∅) → ∅ |
| **draw-switches** | (∅ → ∅) → ∅ |
| **draw-dt-blocks** | (∅ → ∅) → ∅ |
| **draw-time-tbl** | (∅ → ∅) → ∅ |

Tabel 4: Operaties GUI ADT

Het Gui ADT is het ADT die zal zorgen voor het tekenen van de userinterface. De procedure start zorgt ervoor dat wanneer we op een zekere knop drukken, bepaalde evenementen plaatsvinden zoals het openen van een nieuwe frame. De procedure exit zal de frame sluiten, alle threads, die nodig zijn voor het updaten van de berichten, stoppen en andere functies uitvoeren. Daarnaast zorgt de Gui voor het tekenen van de commands voor de trein, switch, detectieblok en om trajecten uit een bestand te kunnen uitvoeren. Deze procedures nemen een callback als argument waarin alle procedures zitten die we nodig hebben wanneer we bijvoorbeeld op een zekere knop drukken.

## Railway ADT

|  |  |
| --- | --- |
| Naam | Signatuur |
| **make-railway** | ∅ → Railway |
| **trains** | ∅ → pair |
| **detection-blocks** | ∅ → pair |
| **switches** | ∅ → pair |
| **reset!** | ∅ → ∅ |
| **add-train!** | symbol symbol symbol → ∅ |
| **train-speed** | symbol → number |
| **train-speed!** | symbol number → ∅ |
| **train-prev-dtb!** | symbol symbol → ∅ |
| **train-direction** | symbol → number |
| **train-inverse-direction!** | symbol → ∅ |
| **train-trajectory?** | symbol → boolean |
| **train-trajectory!** | symbol → ∅ |
| **make-switches!** | pair → ∅ |
| **switch-position** | symbol → number |
| **switch-position!** | symbol, number → ∅ |
| **make-detection-blocks!** | pair → ∅ |
| **dtb-last-train** | symbol → symbol |
| **dtb-last-train!** | symbol, symbol → ∅ |

Tabel 5: Operaties Railway ADT

Het Railway ADT stelt het spoornetwerk voor. Het houdt de volgende zaken bij: de aangemaakte treinen, de wissels en de detectieblokken. Deze kunnen we terugkrijgen door respectievelijk trains, detection-blocks en switches op te roepen. Om deze lijsten weer leeg te maken kunnen we de procedure reset! gebruiken. add-train! zal een nieuw train-object toevoegen aan de lijst van treinen. Om de snelheid van een trein te kennen, gebruiken we de procedure train-speed en om deze te wijzigen gebruiken we train-speed!. Dankzij train-prev-dtb! kunnen we de vorige positie (detectieblok) van een trein wijzigen. Om te weten in welke richting een trein aan het rijden is, roepen we train-direction. Om de richting te veranderen in de tegengestelde richting gebruik je train-inverse-direction!. We kunnen ook weten of een trein een traject aan het uitvoeren is door train-trajectory? op te roepen. Om de waarde, van trajectory? van een trein te wijzigen, roepen we train-trajectory! op. Met make-switches! en make-detection-blocks! maken we de wissels en detectieblokken aan en zetten we die in de lijst. switch-position en switch-position! zorgen ervoor dat we de positie van een wissel kunnen lezen en deze veranderen. Met dtb-last-train en dtb-last-train! kan je weten welke trein het laatst op een bepaalde detectieblok was en deze updaten als het verandert.

## Infrabel

|  |  |
| --- | --- |
| Naam | Signatuur |
| **make-infrabel** | ∅ → Infrabel |
| **trains** | ∅ → pair |
| **dt-blocks** | ∅ → pair |
| **switches** | ∅ → pair |
| **add-train!** | symbol, symbol → ∅ |
| **speed** | symbol → number |
| **speed!** | symbol number → ∅ |
| **train-dt-block** | symbol → symbol |
| **train-direction** | symbol → number |
| **train-inverse-direction!** | symbol → ∅ |
| **train-trajectory?** | symbol → boolean |
| **train-trajectory!** | symbol → ∅ |
| **dtb-last-train** | symbol → symbol |
| **switch-position** | symbol → number |
| **switch-position!** | symbol, number → ∅ |
| **start** | symbol, pair → ∅ |
| **stop** | ∅ → ∅ |
| **add-nmbs!** | Nmbs → ∅ |

*Tabel 6: Operaties Infrabel ADT*

Infrabel is verantwoordelijk voor de communicatie met de simulator/hardware en het aansturen van de verschillende componenten. Het heeft ook een eigen representatie van het spoornetwerk. Het bevat ongeveer dezelfde procedures als in het Railway ADT behalve dat het ook de simulator zal aansturen. trains, dt-block en switches geven een lijst van objecten terug. Met add-train!, speed, speed!, train-dt-block, train-direction en train-inverse-direction! kan je de staat (en de informatie die trein bijhoudt) van een trein veranderen. dtb-last-train geeft de laatste trein die op een bepaalde detectieblok was. Met train-trajectory? Kan je weten of een trein een traject volgt of niet en met train-trajectory! kan je deze waarde wijzigen. switch-position en switch-position! zijn procedures voor de wissels. Meer informatie over deze procedures vind je in 4.5 Railway ADT. In start kunnen we kiezen met welke opstelling we willen beginnen en hierin worden alle elementen aangemaakt. De treinen worden aangemaakt dankzij een lijst van posities dat werd meegegeven. De thread, die zorgt voor het updaten van de ‘laatste trein op een detectieblok’ en ‘laatste detectieblok van een trein’, wordt aangemaakt. Hierin wordt ook de simulator gestart. De procedure stop gaat de lijsten weer leeg maken en de thread stoppen. add-nmbs! zorgt ervoor dat we later NMBS kunnen toevoegen in Infrabel. Wanneer we iets wijzigen in de railway van Infrabel, moet dit ook gebeuren in de railway van NMBS.

## NMBS ADT

|  |  |
| --- | --- |
| Naam | Signatuur |
| **make-nmbs** | Infrabel → Nmbs |
| **start** | ∅ → ∅ |
| **stop** | ∅ → ∅ |
| **railway** | ∅ → Railway |

Tabel 7: Operaties NMBS ADT

Het NMBS ADT zorgt ervoor dat wanneer de gebruiker iets wijzigt in de GUI, dit ook werkelijk gebeurt. Dit is mogelijk door gebruik te maken van callbacks, die opgeroepen worden in de GUI. De callbacks bevatten de procedures om bijvoorbeeld de snelheid van de trein te veranderen, de posities van de wissels te wijzigen… De dynamische snelheid kunnen we terugvinden in de train-callback en het uitlezen van tijdstabellen vinden we terug in het time-tbl-callback. De trein bepaald dynamisch zijn snelheid door te kijken op welk detectieblok het is. Als we naar de opstelling kijken dan zien we dat de trein snel mag reiden op alle detectieblokken die met een 1 beginnen behalve 1-8 (kort stuk spoor), 1-7 en 1-4 (deze zijn bochten). De trein moet traag rijden op alle andere delen omdat het ofwel korte stukken spoor zijn ofwel bochten. Wanneer de trein zich naar een doodlopend stuk spoor verplaatst, dan stopt het. Om een trein een traject, uit een bestand, te laten volgen gebruik ik een loop die lijn per lijn mijn bestand zal lezen. Dit kunnen we terugvinden in de procedure follow-trajectory! Eerst leest het de id van de trein en daarna kan het verder gaan met de volgende instructies. Afhankelijk van wat er staat, zal het bepaalde taken uitvoeren. Bijvoorbeeld: S-2-3/3 zal wissel S-2-3 op positie 3 zetten, 1-1/stop zal de trein laten stoppen wanneer het op detectieblok 1-1 is, +start en -start dienen om de trein weer te starten (in dezelfde of tegengestelde richting) nadat het gestopt werd en finish betekent dat het traject gedaan is. Wanneer een trein een traject aan het uitvoeren is, kan de gebruiker de staat van de trein niet wijzigen. Ik koos voor deze stijl omdat de instructies duidelijk zijn voor de gebruiker en niet zo moeilijk zijn te vertalen voor de programmeur. De procedure start zal de GUI opstarten en de nodige procedures uitvoeren. Bijvoorbeeld: wanneer de gebruiker op “simulator” drukt, dan wordt de simulator geopend. Met de procedure stop, kunnen we (door op de knop “Exit” te drukken) de GUI sluiten. De simulator wordt ook gestopt. Daarnaast heeft NMBS ook een eigen representatie van de railway, hiermee kan het bijvoorbeeld de positie van treinen in de GUI tonen.

## Andere

* + Het bestand helpfunctions.rkt bevat de hulpfuncties get-time en make-id. Deze procedures hebben geen argumenten en hebben als ouput de huidige tijd (als string: “10:20:33”) en een id (als symbol: ‘T-1). Get-time wordt gebruikt in het GUI (voor het logboek). Make-id wordt gebruikt in Infrabel om een nieuwe trein aan te maken.
  + constants.rkt bevat constanten die worden gebruikt/zullen gebruikt worden door verschillende
  + het bestand trajectories bevat de txt-bestanden met trajecten die treinen kunnen uitvoeren. Voorlopig vind je in dit bestand:   
    - trajectory-test.txt: bevat een ‘complexe’ route voor 1 trein.  
    - multiple-trajectories-test.txt: bevat routes voor 3 treinen.  
    - dead-end-test.txt: bevat een test die controleert of een trein wel stopt wanneer het op een doodlopend stuk spoor is.

# Gebruikte bronnen

|  |  |
| --- | --- |
| Onderwerp | Link |
| **GUI** | <https://docs.racket-lang.org/gui/> |
| **Threads** | <https://docs.racket-lang.org/reference/threads.html> <https://docs.racket-lang.org/guide/concurrency.html>  <https://gist.github.com/tfidfwastaken/dbcd525e03b7fb1561342a80f54371a0> |
| **I/O** | <https://docs.racket-lang.org/guide/i_o.html> |
| **Date** | <https://docs.racket-lang.org/reference/time.html> |
| **Structs** | [https://stackoverflow.com/questions/58994448/access-a-struct-field-if-the-](https://stackoverflow.com/questions/58994448/access-a-struct-field-if-the-struct-is-part-of-a-list) [struct-is-part-of-a-list](https://stackoverflow.com/questions/58994448/access-a-struct-field-if-the-struct-is-part-of-a-list) |

*Tabel 8: Gebruikte bronnen*