

Programeerproject 2

Fase 3



Mikhael Swusten

Inhoudstafel

1. Inleiding	2
2. ADT's.....	3
2.1 Train ADT	3
2.2 Railway object ADT	4
2.3 Switch ADT	4
2.4 Barriers ADT	5
2.5 Trafficlight ADT.....	5
2.6 Detectionblock ADT	5
2.7 Railway ADT	6
2.8 GUI ADT.....	6
2.9 Railway-graph ADT	7
2.10 Infrabel ADT	7
2.11 NMBS ADT	8
3. Afhankelijkheidsdiagram	9

1. Inleiding

De inhoud van dit document zal de implementatie omvatten van een controlesysteem voor een modelspoor. Dit zal gerealiseerd worden in de programmeertaal Racket. Hiervoor zullen er een aantal ADT's nodig zijn om het controlesysteem te laten werken. Deze worden later verder beschreven wat het exacte nut van elk type ADT bevat.

In deze fase zullen de volgende functionaliteiten moeten worden voorzien en kunnen worden bestuurd door een grafische gebruikersinterface:

- De locomotieven moet kunnen toevoegen, starten, stoppen, alsook hun snelheid en rij richting veranderen.
- De stand van wissels kunnen uitlezen en verzetten.
- De stand van de lichten kunnen wijzigen.
- De slagbomen kunnen heffen en zakken.
- Detectieblokken kunnen detecteren waar treinen zich bevinden.

2. ADT's

2.1 Train ADT

Het Train ADT maakt een trein aan op het spoor die door de bestuurder kan bestuurd worden.

Naam	Signatuur
train%	(symbol symbol symbol -> train-adt)
get-id	($\emptyset$ -> symbol)
get-direction	($\emptyset$ -> symbol)
get-speed	($\emptyset$ -> number)
get-prev-segment	($\emptyset$ -> symbol)
get-curr-segment	($\emptyset$ -> symbol)
get-vis-segment	($\emptyset$ -> list)
change-direction!	($\emptyset$ -> $\emptyset$)
get-trajectory	($\emptyset$ -> list $\cup$ symbol)
set-speed!	(number -> $\emptyset$)
add-vis-segment!	(symbol -> $\emptyset$)
set-segment!	(symbol -> $\emptyset$)
set-trajectory!	(list -> $\emptyset$)
update-segments!	(symbol symbol symbol -> $\emptyset$)

- De train% maakt het trein adt aan met 3 symbols die staan voor de id, previous segment en de current segment.
- De get-train, get-direction en get-speed operaties geven elk respectievelijk de id, directie en snelheid van de trein.
- De get-prev-segment en get-curr-segment operaties geven de vorige segment en huidige segment. Met de get-vis-segment krijg je een lijst met alle bezochte segmenten van deze trein.
- De change-direction! operatie verandert de directie van de trein.
- De set-speed! operatie verandert de snelheid van de trein met een meegegeven nummer.
- De add-vis-segment! operatie voegt een segment toe aan de bezochte segment lijst.
- De set-segment! operatie verandert het huidige segment.
- De update-segments! operatie verandert de huidige, vorige en bezochte segmenten.
- De get-trajectory operatie geeft het traject van de trein en de set-trajectory! Operatie verandert het traject.

2.2 Railway object ADT

Het Railway object ADT maakt een railway object aan. Deze wordt gebruikt als superclass voor het Switch, Barrier en Trafficlight ADT.

Naam	Signatuur
rail-object%	(symbol, symbol, symbol $\cup$ number, symbol $\cup$ number $\rightarrow$ rail-object-adt)
get-type	($\emptyset \rightarrow$ symbol)
get-state	($\emptyset \rightarrow$ symbol $\cup$ number)
get-id	($\emptyset \rightarrow$ symbol)
change-state!	($\emptyset \rightarrow \emptyset$)

- Het rail-object% maakt een railway object aan met een meegegeven type, id, initiële state en alternatieve state. (Al deze zijn symbols behalve bij het maken van een switch, dan zullen de initiële state en alternatieve state numbers zijn.)
- De get-type, get-state en get-id operaties geven elk respectievelijk de id, state en type van het railway object.
- De change-state! operatie verandert de state van het railway object.

2.3 Switch ADT

Het Switch ADT maakt een schakel aan op het spoor waardoor de trein van spoor kan veranderen. Het Switch ADT "inherit" alle procedures van het Railway object ADT.

Naam	Signatuur
switch%	(symbol $\rightarrow$ rail-object-adt)

- De switch% maakt een railway object aan met een symbol als id.

2.4 Barriers ADT

Het Barrier ADT maakt een slagboom aan op het spoor. Het Barrier ADT "inherit" alle procedures van het Railway object ADT.

Naam	Signatuur
barrier%	(symbol -> rail-object-adt)

- De barrier% maakt een railway object aan met een symbol als id.

2.5 Trafficlight ADT

Het Trafficlight ADT maakt een verkeerslicht aan op het spoor die ervoor zorgt dat de trein wel of niet mag doorrijden. Het Trafficlight ADT "inherit" alle procedures van het Railway object ADT.

Naam	Signatuur
light%	(symbol -> rail-object-adt)

- De light% maakt een railway object aan met een symbol als id.

2.6 Detectionblock ADT

Het Detectionblock ADT maakt een detectieblock aan die aangeeft als een trein op deze block staat.

Naam	Signatuur
detection-block%	(symbol -> detectionblock-adt)
get-id	($\emptyset$ -> symbol)
occupied?	($\emptyset$ -> boolean)
change-occupied!	($\emptyset$ -> $\emptyset$)

- De detection-block% maakt het detectieblock aan met een symbol die als id werkt. Deze id kan met get-id opgeroepen worden.
- occupied? is een boolean die initieel op #f staat en op #t kan gezet worden met change-occupied! wanneer deze block wordt bezocht door een trein.

2.7 Railway ADT

Het Railway ADT maakt heel het spoornetwerk aan bestaande uit switches, detection blocks, barriers, traffic lights en trains.

Naam	Signatuur
make-railway-adt	(list, list, list, list, list -> railway-adt)
get-object-list	(symbol -> list)
update-train!	(symbol symbol list -> $\emptyset$)
update-object!	(symbol symbol symbol -> $\emptyset \cup \text{symbol} \cup \text{number}$)
update-block!	(symbol symbol -> $\emptyset \cup \text{boolean} \cup \text{list}$)

- Het make-railway-adt maakt het spoornetwerk aan met 5 lijsten bestaande uit ids van de switches, detection blocks, barriers, traffic lights en trains.
- Met het juiste symbool kan je met get-object-list een lijst van objecten opvragen.
- De update-train! procedure handelt alle procedures te maken met de trein. Het neemt 2 symbolen als id en een procedure naam van de trein (zoals add-train, remove-train!, ...) en een lijst van argumenten.
- De update-object! procedure handelt alle procedures te maken met de switches, barriers en lights. Het neemt 3 symbolen: id, type van het object (switch, barrier of light) en een procedure naam (zoals get-state of change-state!).
- De update-block! procedure handelt alle procedures te maken met de detectionblocks. Het neemt 2 symbolen: id en een procedure naam (zoals occupied?, change-occupied! of get-occupied).

2.8 GUI ADT

Het GUI ADT maakt de grafische gebruikersinterface waardoor je het spoornetwerk kan beïnvloeden.

Naam	Signatuur
make-gui-adt	(any -> gui-adt)
update-trains!	(list list -> $\emptyset$)

- De make-gui-adt maakt het gui adt aan.
- De update-trains! procedure zorgt ervoor dat de informatie tussen de gui en het spoor van nmbs gesynchroniseerd zijn.

2.9 Railway-graph ADT

Het Railway-graph ADT zorgt voor de communicatie tussen het spoornetwerk en de simulator/modelbouwhardware.

Naam	Signatuur
make-railway-graph	($\emptyset \rightarrow$ make-railway-graph)
getters	(symbol $\rightarrow$ list $\cup$ Boolean $\cup$ symbol $\cup$ number)
setters	(symbol $\rightarrow \emptyset$)
is-switch?	(symbol $\rightarrow$ boolean)
on-legs?	(symbol symbol symbol $\rightarrow$ boolean)

- De make-railway-graph maakt het infrabel adt aan.
- Met de getters procedure kan je verschillende waardes opvragen van de graf of de hashtable.
- Met de setters procedure kan je verschillende waardes veranderen van de hashtable.
- De is-switch? procedure krijgt een symbool binnen en gaat kijken of het een switch is.
- De on-legs? procedure krijgt een 3 symbolen binnen en gaat kijken of 2 segmenten op de benen van een switch staan.

2.10 Infrabel ADT

Het Infrabel ADT zorgt voor de communicatie tussen het spoornetwerk en de simulator/modelbouwhardware.

Naam	Signatuur
make-infrabel-adt	(any $\rightarrow$ infrabel-adt)
close-simulator	($\emptyset \rightarrow \emptyset$)
receive-msg	(list $\rightarrow \emptyset$)

- De make-infrabel-adt maakt het infrabel adt aan.
- De close-simulator procedure zorgt ervoor de simulator uit gaat en dat de update timer stopt.
- De receive-msg zorgt voor het uitvoeren van infrabel procedures op zijn eigen spoornetwerk gekregen van nmbs als lijst.

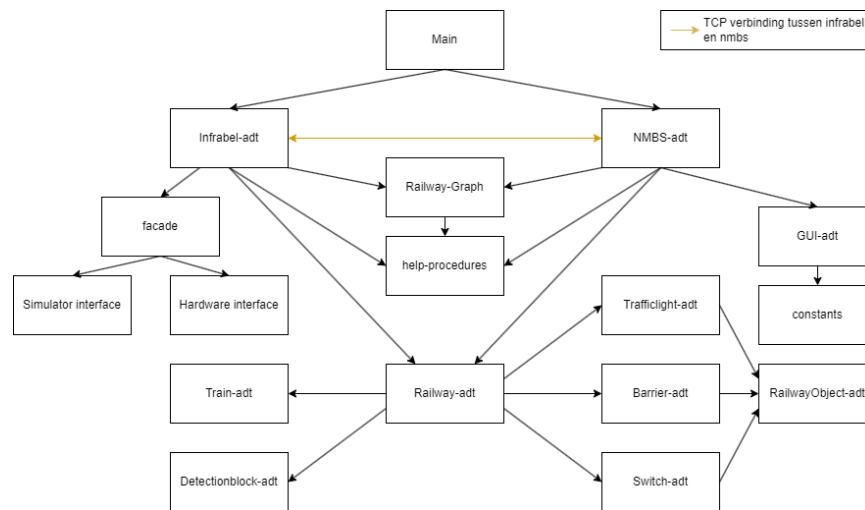
2.11 NMBS ADT

Het NMBS ADT zorgt voor de connectie tussen het spoornetwerk en de gui. Ook zorgt het voor het uitstippelen van trajecten of het opstellen van tijdstabellen.

Naam	Signatuur
make-nmbs-adt	(any -> nmbs-adt)
receive-msg	(list -> $\emptyset$)

- De make-nmbs-adt maakt het nmbs adt aan.
- De receive-msg zorgt voor het uitvoeren van nmbs procedures op zijn eigen spoornetwerk gekregen van infrabel als lijst.

3. Afhankelijkheidsdiagram



- Main.rkt is afhankelijk van infrabel adt en NMBS adt omdat deze twee adts worden aangemaakt en opgeroepen in main.
- Er is een tcp verbinding tussen infrabel en nmbs die het toelaat om berichten naar elkaar te sturen.
- Infrabel adt is afhankelijk van railway adt en facade.rkt omdat in infrabel adt het spoornetwerk geüpdatet wordt en dan grafisch weergegeven wordt door de simulator of uitgevoerd wordt door de hardware-opstelling. Ook is het afhankelijk van Railway-Graph adt om traject af te leggen en reservaties te maken.
- NMBS adt is afhankelijk van railway adt, GUI adt en Railway-Graph adt. NMBS maakt een traject aan met behulp van Railway-Graph adt en stuurt het naar Infrabel. GUI adt beïnvloedt het spoornetwerk van nmbs.
- Facade.rkt is afhankelijk van de interfaces van de hardware en van de simulator. Dit bestand zal de juiste interface gebruiken afhankelijk van de gekozen optie.
- Het railway adt is afhankelijk van het train, switch, detectionblock, barriers en trafficlight adt want hier wordt het gehele spoornetwerk gevormd en bewaard aan de hand van al deze adts.
- Switch, barriers en trafficlight adt zijn afhankelijk van railway object adt want deze drie gebruiken railway object adt als super class bij aanmaak.