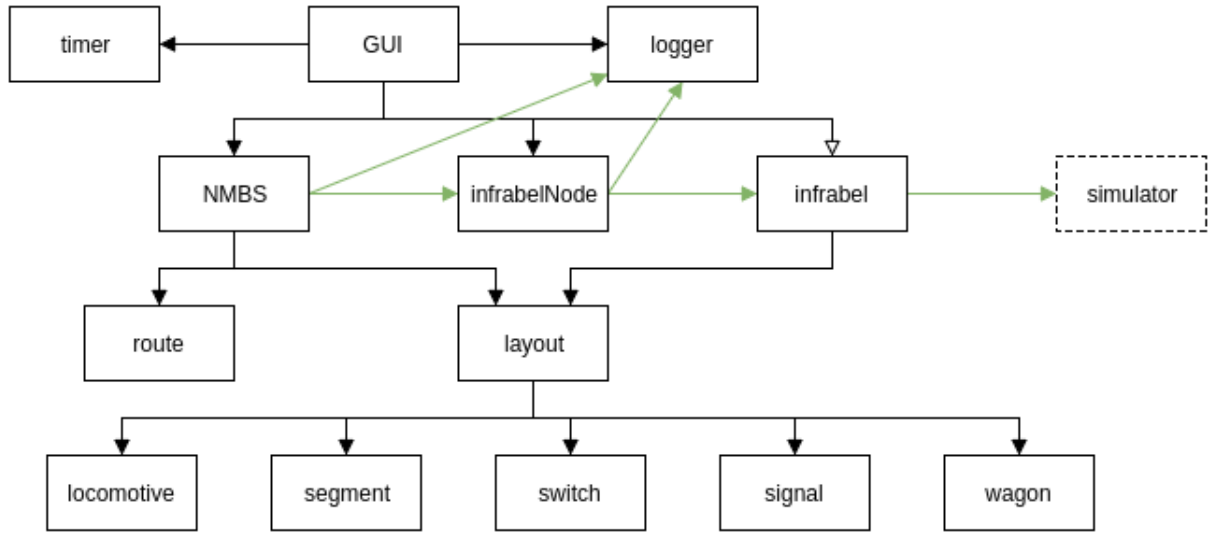
# Programmeerproject 2: Voorstudie

Jannick Hemelhof

30 oktober 2016

# Inhoudsopgave

# 1 Overzichtsdiagram




Figuur 1: Overzichtsdiagram

De pijlen hebben volgende betekenis:

- zwarte pijl met gevulde punt "x → y": Dit wil zeggen dat x gebruikt maakt van y, x doet dus beroep op de 'make'-functie van y en vervolgens met dit object zal werken (er mee communiceren).
- groene pijl met gevulde punt "x → y": Dit wil zeggen dat x enkel en alleen communiceert met y, het geeft dus aan hoe de belangrijkste lijnen van de communicatie lopen.
- zwarte pijl met lege punt "x → y": Deze pijl heeft als betekenis dat x het object y alleen maar zal aanmaken. Het wordt eventueel meegegeven aan andere objecten die x zal aanmaken maar er wordt niet (rechtstreeks) gecommuniceerd met y.

# 2 Bespreking ADTs

## 2.1 route

Een ADT dat instaat voor het beheer van een route over meerdere segmenten en/of wissels heen. Houdt bij welke segmenten we moeten bereiken om vanuit onze origin naar de destination te geraken. Bij aanmaak worden twee argumenten meegegeven met een optioneel derde argument:

- origin: Van waar vertrekt deze route?
- destination: Naar waar gaat deze route?
- path: Optioneel derde argument in de vorm van een lijst met te volgen segmenten en/of wissels. Geeft aan welk pad de route exact moet volgen.

Indien we geen derde argument meegeven *kan* er gebruik worden gemaakt van een optimaal pad dat wordt gezocht tussen origin en destination. Volgende boodschappen worden door het ADT ondersteund:

| Commando | Argument(en) | Return value | Omschrijving |
|---|---|---|---|
| getOrigin | / | origin (string) | Geeft de oorsprong van de route terug in de vorm van een string. |
| getDestination | / | destination (string) | Geeft de bestemming van de route terug in de vorm van een string. |
| getLength | / | length (int) | Geeft de lengte van de route terug. |
| detected! | / | / | Geeft aan dat we het volgende detectieblok voorbij zijn en dat we dit hier ook moeten aanpassen. Op deze manier weten we hoe ver we gevorderd zijn op de route en kunnen we indien nodig een trein die het kortst bij zijn bestemming is een hogere prioriteit geven bij een obstakel/ongeval. |
| getTodo | / | todo (list) | Zorgt ervoor dat we een lijst terugkrijgen met daarin de route tot aan het volgende detectieblok. |

Met het *getTodo* commando kan het NMBS-gedeelte eenvoudig bepalen welke commando's naar het Infrabel-gedeelte moeten gestuurd worden om de route voor een zekere trein mogelijk te maken.

## 2.2 timer

Dankzij dit ADT kunnen we de notie van tijdsevolutie gebruiken in het project. Bij aanmaak worden drie argumenten meegegeven:

- timerLength: Hoelang later we onze timer steeds lopen?
- object: Naar welk object zullen we de boodschap sturen telkens de timer afloopt?
- message: Welke boodschap sturen we naar het object als de timer telkens de timer afloopt?

Een timer volgt dus volgend principe:

1. Timer wordt aangemaakt. Is nog niet actief.
2. Timer ontvangt boodschap 'start'
3. Timer begint te lopen vanaf timerLength tot 0
4. 0 wordt bereikt, stuur message naar het object. Ga opnieuw naar 3.

Dit wordt steeds herhaald tot de timer de boodschap 'stop' ontvangt. Volgende boodschappen worden door het ADT ondersteund:

| Commando | Argument(en) | Return value | Omschrijving |
|---|---|---|---|
| start | / | / | Laat de timer starten |
| stop | / | / | Laat de timer stoppen |
| setTimerLength! | timerLength (int) | / | Past de lengte van de timer aan naar de meegegeven lengte. |
| setObject! | object (object) | / | Stuur de boodschap naar een ander object, zijnde het meegegeven object. |
| setMessage! | message (string) | / | Stuur een andere boodschap naar het object, zijnde de meegegeven boodschap. |

## 2.3 signal

Dit ADT stelt een licht voor dat zich kan bevinden op een segment. Het houdt een status bij en stelt andere objecten in staat om deze status op te vragen en aan te passen. Bij aanmaak wordt één argument meegegeven:

- id: Uniek ID dat het licht identificeert

Volgende boodschappen worden door het ADT ondersteund:

| Commando | Argument(en) | Return value | Omschrijving |
|---|---|---|---|
| getID | / | id (string) | Geeft het ID van het licht terug in de vorm van een string. |
| getStatus | / | status (string) | Geeft de status van het licht terug in de vorm van een string. |
| setStatus! | status (string) | / | Past de status van het licht aan naar de meegegeven status. |

## 2.4 logger

Een ADT dat zal instaan voor het loggen - in dit geval wegschrijven naar een bestand - van de belangrijke gebeurtenissen die plaatsvinden tijdens het uitvoeren van de simulatie. Bij aanmaak worden geen argumenten meegegeven. Volgende boodschappen worden door het ADT ondersteund:

| Commando | Argument(en) | Return value | Omschrijving |
|---|---|---|---|
| addTrain | id (string) | / | Logt dat er een nieuwe trein 'id' is toegevoegd. |
| segmentPassed | trainID (string) + segmentID (string) | / | Logt dat een trein 'trainID' een segment 'segmentID' is gepasseerd. |
| switchPassed | trainName (string) + switchName (string) | / | Logt dat een trein 'trainID' een wissel 'switchID' is gepasseerd. |
| speedChanged | trainID (string) + newSpeed (number) | / | Logt dat een trein 'trainID' vertraagt/versnelt tot snelheid 'newSpeed'. |
| directionChanged | trainID (string) + newDirection (number) | / | Logt dat een trein 'trainID' naar richting 'newDirection' is veranderd. |
| trainStopped | trainID (string) + reason (string) | / | Logt een een trein 'trainID' gestopt is vanwege 'reason'. |
| switchChanged | switchID (string) + newPosition (number) | / | Logt dat een switch 'switchID' schakelt naar positie 'newPosition'. |
| signalChanged | signalID (string) + newStatus (string) | / | Logt dat een signaal 'signalID' een nieuwe status 'newStatus' heeft. |
| messageSend | message (string) | / | Logt dat er een commando werd verstuurd naar Infrabel |
| getLogged | / | logged (list) | Geeft een lijst terug met daarin alle acties die werden gelogd sinds de laatste keer dat dit commando werd opgeroepen. |

Een apart commando is beschikbaar in de logger om aan te geven dat er een bericht werd verstuurd naar Infrabel. Aangezien in deel 2 deze via het netwerk moet bereikt worden is het handig om te weten of er wel degelijk een commando over het netwerk werd gestuurd.

## 2.5 locomotive

Het ADT dat een locomotief zal voorstellen en ook zal bijhouden welke wagons er eventueel aan vasthangen. Bij aanmaak worden twee argumenten meegegeven met een optioneel derde:

- id: Een uniek ID voor de trein.
- position: Beginpositie van de trein (een ID van een segment of wissel).
- wagons: Optioneel derde argument dat de wagons vastgemaakt aan de locomotief bevat.

Volgende boodschappen worden door het ADT ondersteund:

| Commando | Argument(en) | Return value | Omschrijving |
|---|---|---|---|
| getID | / | id (string) | Geeft het unieke ID van de locomotief terug in de vorm van een string. |
| getSpeed | / | speed (int) | Geeft de snelheid van de locomotief terug in de vorm van een integer. |
| setSpeed! | newSpeed (int) | / | Past de snelheid van de locomotief aan naar de meegegeven snelheid. |
| getLastSeen | / | lastSeen (string) | Geeft het segment (detectieblok) terug waar de locomotief voor het laatst werd gezien. |
| updateLastSeen! | lastSeen (string) | / | Past aan waar de locomotief voor het laatst werd gezien. |

## 2.6 wagon

Een ADT dat een wagon voorsteld. Een wagon heeft enkel een uniek ID alsook een type dat aangeeft wat de wagon precies vervoert. Bij aanmaak worden twee argumenten meegegeven:

- id: Een uniek ID voor de wagon.
- type: Wat de wagon precies vervoert.

Volgende boodschappen worden door het ADT ondersteund:

| Commando | Argument(en) | Return value | Omschrijving |
|---|---|---|---|
| getID | / | id (string) | Geeft het ID van de wagon terug in de vorm van een string. |
| getType | / | type (string) | Geeft het type van de wagon terug in de vorm van een string. |
| setType! | newType (string) | / | Past het type van de wagon aan naar het meegegeven type. |
| getLoad | / | amount (int) | Geeft aan hoezeer de wagon gevult is |
| load! | amount (int) | / | Vult de wagon met een bepaalde hoeveelheid ('amount') van zijn type |
| unload! | amount (int) | / | Haalt een bepaalde hoeveelheid ('amount') uit de wagon |

## 2.7 switch

Dit ADT stelt een wissel voor. Er wordt bijgehouden welke segmenten met de wissel bereikbaar zijn, wat de huidig status is maar ook eventueel in welke volgorde een schakeling moet gebeuren. Bij aanmaak worden drie argumenten meegegeven met een optioneel vierde:

- id: Een uniek ID voor de wissel.
- segments: Een lijst (minimale lengte 3) met de mogelijke segmenten die bereikbaar zijn met deze wissel.
- default: Welke status wordt gezien als de "default"(d.w.z. conceptueel gezien rechtdoor)?
- switchOrder: Optioneel vierde argument dat aangeeft hoe er tussen de verschillende mogelijke statussen geschakeld moet worden.

Volgende boodschappen worden door het ADT ondersteund:

| Commando | Argument(en) | Return value | Omschrijving |
|---|---|---|---|
| getID | / | id (string) | Geeft het ID van de wissel terug in de vorm van een string. |
| getSegments | / | segments (list) | Geeft alle bereikbare segmenten hun ID terug in de vorm van een lijst. |
| getActive | / | active (list) | Geeft de huidig actieve status terug in de vorm van een lijst met daarin de IDs van bereikbare segmenten. |
| setActive! | mode (list) | / | Laat de wissel schakelen naar de gegeven mode. |

## 2.8 segment

Een ADT dat een normaal treinsegment zonder wissels voorstelt. Er wordt bijgehouden welke segmenten bereikbaar zijn en of er eventueel lichten en een detectieblok op het segment aanwezig zijn. Bij aanmaak worden twee argumenten meegegeven met een optioneel derde en vierde:

- id: Een uniek ID voor het segment.
- segments: Een lijst met de bereikbare segmenten.
- signal: Optioneel derde argument in de vorm van een signal-object dat aangeeft of er een licht op het segment aanwezig is.
- detection: Optioneel vierde argument dat aangeeft of er een detectieblok op dit segment aanwezig is.

Volgende boodschappen worden door het ADT ondersteund:

| Commando | Argument(en) | Return value | Omschrijving |
|---|---|---|---|
| getID | / | id (string) | Geeft het ID van het segment terug in de vorm van een string. |
| getSegments | / | segments (list) | Geeft de bereikbare segmenten terug in de vorm van een lijst. |
| signalCommand | command (string) + extra (list) | result (string) | Zal het licht een bepaalt commando 'command' laten uitvoeren, eventueel met behulp van optionele paramaters 'extra' |

## 2.9 infrabel

ADT dat specifiek instaat voor de infrastructuur, d.w.z. het aansturen van wissels en signalisatie maar ook zorgt voor het respecteren van de veiligheid (afstand tussen treinen bijvoorbeeld). Bij aanmaak worden geen argumenten meegegeven. Volgende boodschappen worden door het ADT ondersteund:

| Commando | Argument(en) | Return value | Omschrijving |
|---|---|---|---|
| addTrain | id (string) + start (string) (+ wagons (list)) | / | Voegt een trein toe. |
| manipulateTrain | id (string) + command (string) + extra (list) | / | Zal een trein 'id' een bepaalt commando 'command' laten uitvoeren, eventueel aan de hand van optionele paramaters 'extra'. |
| manipulateSignal | id (string) + newStatus (string) | / | Zal de status van een signaal 'id' veranderen naar de meegegeven status 'newStatus'. |
| manipulateSwitch | id (string) + newMode (int) | / | Zal een wissel laten schakelen naar de nieuwe modus 'newMode'. |
| getActions | / | actions (list) | Geeft een lijst terug met daarin alle uitgevoerde acties sinds de laatste keer dat dit commando werd opgeroepen. |
| start | / | / | Zal het monitoren inschakelen. |
| stop | / | / | Zal het monitoren uitschakelen. |

## 2.10 infrabelNode

Dit ADT zal instaan voor de communicatie tussen het NMBS-gedeelte en het Infrabel-gedeelte van het programma. Hoewel in Deel 1 deze nog op hetzelfde systeem draaien is dit niet meer het geval in Deel 2 en daar willen we op voorbereid zijn. Conceptueel is het niet logisch dat het NMBS-gedeelte ook zou moeten zorgen voor netwerkcommunicatie, dus vandaar dat deze logica in een apart ADT wordt gestoken. Het ADT accepteert dezelfde commando's als het Infrabel-gedeelte. Bij aanmaak wordt één argument meegegeven:

- logger: Een logger die het ADT zal gebruiken.

Volgende boodschappen worden door het ADT ondersteund:

| Commando | Argument(en) | Return value | Omschrijving |
|---|---|---|---|
| addTrain | id (string) + start (string) (+ wagons (list)) | / | Voegt een trein toe. |
| manipulateTrain | id (string) + command (string) + extra (list) | / | Zal een trein 'id' een bepaalt commando 'command' laten uitvoeren, eventueel aan de hand van optionele paramaters 'extra'. |
| manipulateSignal | id (string) + newStatus (string) | / | Zal de status van een signaal 'id' veranderen naar de meegegeven status 'newStatus'. |
| manipulateSwitch | id (string) + newMode (int) | / | Zal een wissel laten schakelen naar de nieuwe modus 'newMode'. |
| getActions | / | actions (list) | Geeft een lijst terug met daarin alle uitgevoerde acties sinds de laatste keer dat dit commando werd opgeroepen. |
| start | / | / | Zal het monitoren inschakelen. |
| stop | / | / | Zal het monitoren uitschakelen. |

## 2.11 NMBS

ADT dat instaat voor beheer van routes. Zal voor elk locomotief een route bijhouden en op deze manier bepalen welke wissels moeten schakelen. Bij aanmaak wordt één argument meegegeven:

- logger: Een logger die het ADT zal gebruiken.

Volgende boodschappen worden door het ADT ondersteund:

| Commando | Argument(en) | Return value | Omschrijving |
|---|---|---|---|
| createRoute | id (string) + start (string) + destination (string) (+ segments (list)) | / | Wijst aan een trein 'id' een route toe tussen 'start' en 'destination' die eventueel gebruik maakt van bepaalde segmenten/wissels. |
| getDrawable | / | drawable (list) | Geeft een weergave van de opstelling terug die de GUI kan tekenen. |

## 2.12 layout

Een ADT dat onze virtuele opstelling zal bijhouden. Encapsuleert dus een weergave van het veld, een graf van segmenten/wissels en de treinen die er op rijden. Bij aanmaak worden geen argumenten meegegeven. Volgende boodschappen worden door het ADT ondersteund:

| Commando | Argument(en) | Return value | Omschrijving |
|---|---|---|---|
| addTrain | id (string) + start (string) (+ wagons (list)) | / | Voegt een trein toe. |
| manipulateTrain | id (string) + command (string) + extra (list) | / | Zal een trein 'id' een bepaalt commando 'command' laten uitvoeren, eventueel aan de hand van optionele paramaters 'extra'. |
| manipulateSignal | id (string) + newStatus (string) | / | Zal de status van een signaal 'id' veranderen naar de meegegeven status 'newStatus'. |
| manipulateSwitch | id (string) + newMode (list) | / | Zal een wissel laten schakelen naar de nieuwe modus 'newMode'. |
| getDrawable | / | drawable (list) | Zal een lijst teruggegeven met de opbouw van het veld zodat de GUI het kan tekenen. |
| getActions | / | actions (list) | Geeft een lijst terug met daarin alle uitgevoerde acties sinds de laatste keer dat dit commando werd opgeroepen. |
| processActions | actions (list) | / | Past een lijst met acties toe op de opstelling. |
| loadSchema | filename (string) | / | Laadt een opstelling vanuit het bestand 'filename'. |
| saveSchema | / | / | Schrijft de opstelling weg naar een bestand. |
| getIDs | / | IDs (list) | Geeft de IDs terug van alle onderdelen (segmenten, wissels en locomotieven) terug. |

### 2.13 GUI

ADT dat de interface en eigelijk het hele programma zal beheren. Bij aanmaak worden twee argumenten meegegeven:

- height: Hoe hoog moet het venster zijn?
- width: Hoe breed moet het venster zijn?

Volgende booschappen worden door het ADT ondersteund:

| Commando | Argument(en) | Return value | Omschrijving |
|---|---|---|---|
| draw | rate (int) | / | Zorgt dat het venster elke 'rate' seconden opnieuw zal worden getekend. |

De GUI zal dus met behulp van een timer elke rate seconden het beeld refreshen (opnieuw tekenen). Daarvoor moet er natuurlijk eerst bepaalt worden wat er moet worden getekend. Dat zal de GUI aan het NMBS-gedeelte (verantwoordelijk voor de lay-out van de opstelling) vragen met behulp van het *getDrawable* commando. Met de lijst die als return value terugkomt kan de GUI dan verder.

Verder wordt ook user input gedetecteerd en vervolgens verwerkt. Commando's worden verstuurd naar hun verantwoordelijke ADTs (NMBS of infrabelNode). Resultaten van deze commando's worden getoond (= getekend) met de volgende refresh.

## 3 Planning

| Week | Omschrijving |
|---|---|
| 4 | Design ADTs |
| 5 | Design ADTS + schrijven voorstudie |
| 6 | Design ADTs, werken aan ADT wagon + finaliseren voorstudie |
| 7 | Afwerken ADTs timer, locomotive, wagon, signal |
| 8 | Afwerken ADTs segment, switch, route |
| 9 | Afwerken ADTs logger, layout |
| 10 | Werken aan ADTs NMBS, infrabel, infrabelNode |
| 11 | Afwerken ADTs NMBS, infrabel, infrabelNode |
| 12 | Basis van GUI en communicatie tussen GUI en NMBS/infrabelNode |
| 13 | GUI |
| 14 | GUI afwerken |
| 15 | Nakijken code + werken aan verslag |
| 16 | Finaliseren verslag |