Programmeerproject 2: Voorstudie

Jannick Hemelhof

October 29, 2016

Contents

1 Overzichtsdiagram

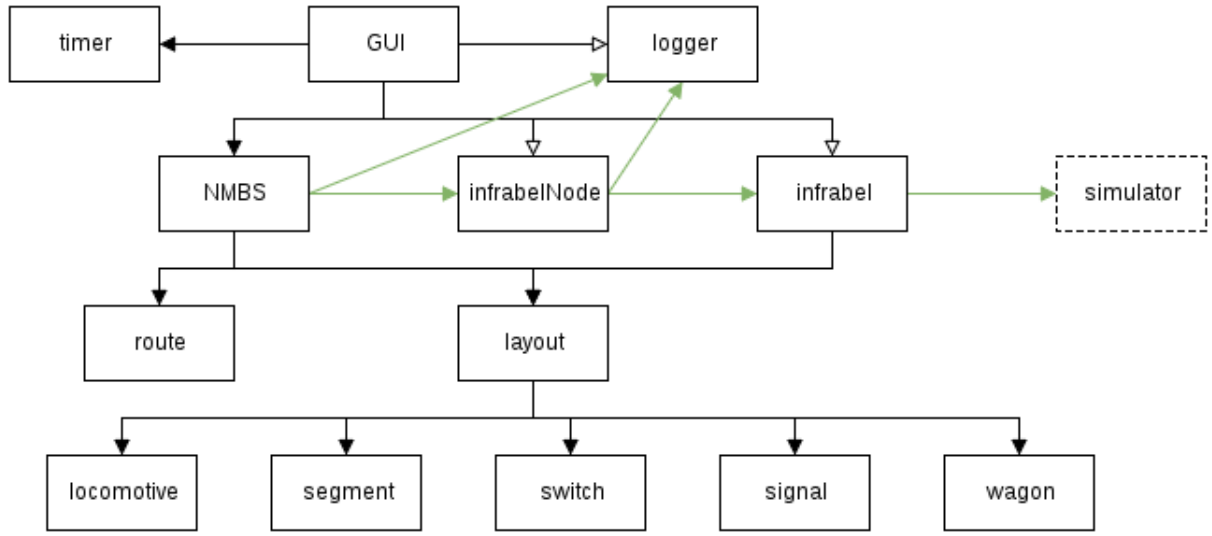

Figure 1: Overzichtsdiagram

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 (number)	Geeft de lengte van de route terug in de vorm van een number.
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 (number)	/	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.
trainStop	trainID (string) + reason (string)	/	Logt een een trein 'trainID' gestopt is vanwege 'reason'.
switchChange	switchID (string) + newPosition (number)	/	Logt dat een switch 'switchID' schakelt naar positie 'newPosition'.
signalChange	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

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. Zal zelfstandig zijn positie updaten aan de hand van het segment of de wissel waar hij zich op bevindt en zijn snelheid. Houdt ook de aan de locomotief bevestigde wagons bij. Bij aanmaak worden vijf argumenten meegegeven:

- name: Een naam voor de trein.
- id: Een uniek ID voor de trein.
- length: Geeft aan hoeveel wagons er zullen hangen aan de locomotief.
- speed: De beginsnelheid van de trein.
- position: Beginpositie van de trein.

Volgende boodschappen worden door het ADT ondersteund:

Commando	Argument(en)	Return value	Omschrijving
getName	/	name (string)	Geeft de naam van de trein terug in de vorm van een string.
getID	/	id (string)	Geeft het unieke ID van de trein terug in de vorm van een string.
getLength	/	length (number)	Geeft de lengte van de trein terug in de vorm van een number.
getSpeed	/	speed (number)	Geeft de snelheid van de trein terug in de vorm van een number.
setSpeed!	newSpeed (number)	/	Past de snelheid van de trein aan naar de meegegeven snelheid.
getPosition	/	position (position)	Geeft de huidige positie van de trein terug in de vorm van een positie.
updatePosition!	/	/	Zorgt dat de trein zijn positie zelfstandig zal updaten aan de hand van waarop hij zich momenteel bevindt alsook zijn snelheid.

Een belangrijk detail in verband met de positie: de positie wordt niet aan de simulator gevraagd, wel aan de trein zelf die deze onafhankelijk bijwerkt. Op deze manier komt het al conceptueel overeen met fase 2 en het werken met een modelopstelling.

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.

2.7 switch

Dit ADT stelt een wissel voor. Er wordt bijgehouden welke segmentRoutes binnenin de wissel mogelijk zijn, wat de huidig segmentRoute is (er kan er maar n actief zijn) maar ook eventueel in welke volgorde een schakeling tussen verschillende segmentRoutes moet gebeuren. Bij aanmaak worden vijf argumenten meegegeven met een optioneel zesde:

- name: Een naam voor de wissel.
- id: Een uniek ID voor de wissel.
- layout: Een layout-ID dat de GUI kan gebruiken om te bepalen hoe de wissel getoond moet worden.
- segmentRoutes: Een lijst (minimale lengte 2) met de mogelijke segmentRoutes.
- default: Welke segmentRoute wordt gezien als de "default" (d.w.z. conceptueel gezien rechtdoor)?
- switchOrder: Optioneel zesde argument dat aangeeft hoe er tussen de verschillende mogelijke segmentRoutes geschakeld moet worden.

Volgende boodschappen worden door het ADT ondersteund:

Commando	Argument(en)	Return value	Omschrijving
getName	/	name (string)	Geeft de naam van de wissel terug in de vorm van een string.
getID	/	id (string)	Geeft het ID van de wissel terug in de vorm van een string.
getLayout	/	layout (string)	Geeft de layout van de wissel terug in de vorm van een string.
getActive	/	active (segmentRoute)	Geeft de huidig actieve segmentRoute terug in de vorm van een segmentRoute.
setActive!	mode (int)	/	Laat de wissel schakelen naar de gegeven mode.

2.8 segment

Een ADT dat een normaal treinsegment zonder wissels voorstelt. Er wordt bijgehouden welke segmentRoutes binnen het segment mogelijk zijn. Bij aanmaak worden 4 argumenten meegegeven:

- name: Een naam voor het segment.
- id: Een uniek ID voor het segment.
- layout: Een layout-ID dat de GUI kan gebruiken om te bepalen hoe het segment getoond moet worden.
- segmentRoutes: Een lijst met de mogelijke segmentRoutes.

Volgende boodschappen worden door het ADT ondersteund:

Commando	Argument(en)	Return value	Omschrijving
getName	/	name (string)	Geeft de naam van het segment terug in de vorm van een string.
getID	/	id (string)	Geeft het ID van het segment terug in de vorm van een string.
getLayout	/	layout (string)	Geeft de layout van het segment terug in de vorm van een string.
getSegmentRoute	destination (string)	segmentRoute (segmentRoute)	Geeft de segmentRoute naar de destination terug in de vorm van een segmentRoute.

2.9 controlSystem

Dit ADT is een ADT dat het infrastructuur- en beheergedeelte overkoepelt. Het zal er dan ook voor zorgen dat de boodschappen die het ontvangt worden doorgespeeld aan het conceptueel verantwoordelijke ADT. De reden waarom het op deze manier wordt gedaan is zodat wanneer de verantwoordelijkheid verandert, bv. de nmbs wordt verantwoordelijk voor de wissels, dan moet er niets veranderen bij objecten die wissels willen manipuleren. Zij geven gewoon aan dat ze dit willen doen aan het overkoepelde ADT en dit maakt dan ook uit wie verantwoordelijk is. Bij aanmaak wordt 1 argument meegegeven:

- logger: Een logger die het controlesysteem zal gebruiken.

Volgende boodschappen worden door het ADT ondersteund:

Commando	Argument(en)	Return value	Omschrijving
addTrain	speed (number) + length (number) + pos (position) + name (string) + id (string)	/	Voegt een trein toe die zal worden beheerd door het controlSystem.
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' indien dit niet zorgt voor het onmogelijk maken van bepaalde routes.
manipulateSegment	id (string) + maxSpeed (int)	/	Zal een segment/wissel 'id' een maximumsnelheid 'maxSpeed' opleggen.
setRoute	id (string) + destination (string)	/	Zal een trein 'id' een route opleggen met bestemming 'destination'.
start	/	/	Zal het controlesysteem inschakelen.
stop	/	/	Zal het controlesysteem uitschakelen.

2.10 infrastructure

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	loco (locomotive)	/	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' indien dit niet zorgt voor het onmogelijk maken van bepaalde routes.
manipulateSegment	id (string) + maxSpeed (int)	/	Zal een segment/wissel 'id' een maximumsnelheid 'maxSpeed' opleggen.
start	/	/	Zal het monitoren van de infrastructuur inschakelen.
stop	/	/	Zal het monitoren van de infrastructuur uitschakelen.

Alle manipulaties vertrekken vanuit dit ADT naar de simulator indien het een geldig commando is dat niet in conflict is met eerder uitgevoerde manipulaties en de geplande routes niet verstoord. Dit wordt gedaan met in het achterhoofd fase 2 omdat we zo onnodige communicatie met een extern apparaat vermijden indien we op voorhand al weten dat het commando niet zal worden uitgevoerd.

2.11 management

ADT dat instaat voor beheer van routes. Zal voor elk trein-ID een route bijhouden en op deze manier het infrastructuurgedeelte toestaan te bepalen welke wissels moeten schakelen. Bij aanmaak worden geen argumenten meegegeven. Volgende boodschappen worden door het ADT ondersteund:

planRoute	start (string) + destination (string)	route	Plant een route tussen 'start' en 'destination' en geeft deze dan terug in de vorm van een route.
createRoute	start (string) + destination (string) + segments (list)	route	Maakt een route aan tussen 'start' en 'destination' die gebruik maakt van bepaalde segments/wissels en geeft deze dan terug in de vorm van een route.
assignRoute	id (string) + route (route)	/	Wijst aan een trein 'id' een bepaalde route 'route' toe.
getPosition	id (string)	pos (position)	Geeft terug waar een trein 'id' zich precies op zijn route bevindt in de vorm van een position.

2.12 simulator

Een ADT dat onze virtuele opstelling zal bijhouden. Encapsuleert dus een weergave van het veld, een raster van segmenten/wissels. Bij aanmaak worden geen argumenten meegegeven. Volgende boodschappen worden door het ADT ondersteund:

Commando	Argument(en)	Return value	Omschrijving
addTrain	loco (locomotive)	/	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'.
manipulateSegment	id (string) + maxSpeed (int)	/	Zal een segment/wissel 'id' een maximumsnelheid 'maxSpeed' opleggen.
getLayout	/	layout (list)	Zal een lijst teruggegeven met de opbouw van het veld zodat we het kunnen tekenen.
start	/	/	Zal de simulator starten.
stop	/	/	Zal de simulator stoppen.

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	/	/	Zorgt dat het venster zal worden getekend.

3 Planning

Week	Omschrijving
6	Design ADTs + werken aan verslag
7	Finaliseren verslag + bouwen van ADT timer
8	Afwerken ADTs locomotive, wagon, signal, position
9	Afwerken ADTs segment, switch, route, segmentRoute
10	Afwerken ADTs controlSystem, logger, infrastructure, management
11	Afwerken simulator + communicatie tussen simulator en infrastructure/management
12	Basis van GUI en communicatie tussen GUI en simulator/controlSystem
13	GUI
14	GUI afwerken
15	Nakijken code + werken aan verslag
16	Finaliseren verslag