Metro Simulatie

Documentsoort:	Behoeftespecificatie
Versie:	2.0
Datum:	28-03-2023
Auteurs:	Kasper Engelen

1 Samenvatting

Dit document bevat de specificaties voor een informaticasysteem ter ondersteuning van een metro simulatie. Het is geschreven in het kader van het vak “Project Software Engineering” (1ste bachelor informatica - Universiteit Antwerpen).

2 Context

Sinds 30 januari maakt de gewestelijke vervoersmaatschappij De Lijn in de stad Antwerpen gebruik van een nieuw type tram: de Stadslijner. De PCC-trams uit de jaren 60 en 70 zijn verouderd, wat leidt tot defecten en reparatiekosten. Het is daarom dat De Lijn besloten heeft om een nieuw type trams aan te kopen. Om dit alles in goede banen te leiden, heeft De Lijn ervoor gekozen om een simulatietool te laten bouwen om het tramverkeer, en de onderhoudskosten van de PCC-trams, in kaart te brengen.

De Universiteit Antwerpen is gevraagd dit systeem te ontwikkelen. In de eerste bachelor informatica zal onder de vakken “Project Software Engineering” en “Computer Graphics” gewerkt worden aan dit project. Tijdens de practica Project Software Engineering zal gewerkt worden aan de simulatie applicatie zelf, tijdens de practica Computer Graphics zal de visualisatie van de simulatie ontwikkeld worden.

3 Legende

De behoeftespecificatie is opgesteld aan de hand van zogenaamde use-cases. Elke use-case beschrijft een klein gedeelte van de gewenste functionaliteit. Het is de bedoeling dat tijdens elke fase van het project verschillende use cases geïmplementeerd worden. Een typische use-case bevat de volgende onderdelen:

- **Use case nummer & titel:**
  Wordt gebruikt om naar een bepaalde use-case te verwijzen.
- **Prioriteit:**
  De specificatie van een systeem bevat meer use-cases dan wat binnen de voorziene tijd kan worden afgewerkt. Vandaar dat we per use-case aangeven in hoeverre die functionaliteit belangrijk is. In volgorde van belangrijkheid kan hier staan: VERPLICHT (deze use-case moet opgeleverd worden), BELANGRIJK (niet essentieel maar bij voorkeur toch opleveren), NUTTIG (interessant maar kan weggelaten worden).
- **Doel:**
  Summiere beschrijving van het waarom van de use-case, t.t.z. wat de use-case bijdraagt tot de gehele functionaliteit.
- **Preconditie:**
  Summiere beschrijving van de voorwaarden waaraan voldaan moet worden vooraleer de use-case in werking kan treden.
- **Succesvol einde:**
  Summiere beschrijving van wat de use-case zal doen als er niks fout is gegaan.
- **Stappen:**
  Een sequentiële beschrijving van hoe de use-case precies zal verlopen als alles goed gaat (het zogenaamde “happy day scenario”). De stappen zijn genummerd en kunnen controle instructies (WHILE, IF, ...) bevatten.
- **Uitzonderingen:**
  Een lijst van mogelijke probleemgevallen en hoe die behandeld zullen worden. Een probleem geval zal de volgende zaken bevatten:
  (a) een verwijzing naar het nummer van de stap waar het probleem kan optreden,
  (b) een conditie die aangeeft wanneer het probleemgeval optreedt,

(c) een heel korte beschrijving (één lijn) van hoe het probleem behandeld zal worden.

- **Voorbeeld:**
  Een voorbeeld van de invoer of uitvoer van de use-case.

Soms is een use-case een uitbreiding van een andere use-case, en dan zijn volgende onderdelen relevant:

- **Uitbreiding:**
  Een verwijzing naar de use-case waarvan deze een uitbreiding is.
- **Stappen:**
  Een lijst van extra en/of aangepaste stappen t.o.v de use-case waarvan deze een uitbreiding is. Een uitbreiding vermeldt de volgende zaken:

  (a) een verwijzing naar het nummer van de stap die uitgebreid wordt,

  (b) of de uitbreiding voor, na of tijdens de normale stap zal gebeuren,

  (c) een beschrijving van wat precies in de uitbreiding zal gebeuren.

4 Overzicht

Use-Case	**Prioriteit**
1: Invoer	
Use case 1.1: Trams en stations inlezen	VERPLICHT
Use case 1.2: Trams en stations met type inlezen	VERPLICHT
Use case 1.3: Trams met voertuignummers	BELANGRIJK
Use case 1.4: Stations met meerdere sporen inlezen	NUTTIG
2: Uitvoer	
~~Use case 2.1: Simpele uitvoer (oude versie)~~	~~VERPLICHT~~
Use case 2.2: Simpele uitvoer (herzien)	VERPLICHT
Use case 2.3: Geavanceerde uitvoer	BELANGRIJK
Use case 2.4: Grafische 3D rendering	BELANGRIJK
3: Simulatie	
Use case 3.1: Rijden van trams	VERPLICHT
~~Use case 3.2: Automatische simulatie (oude versie)~~	~~VERPLICHT~~
Use case 3.3: Automatische simulatie (herzien)	VERPLICHT
Use case 3.4: Rijden van trams met type	VERPLICHT
Use case 3.5: Rijden van meerdere trams	VERPLICHT
Use case 3.6: Botspreventie	BELANGRIJK
Use case 3.7: Defecten en reparaties	VERPLICHT
Use case 3.8: Reparatiekosten	BELANGRIJK
Use case 3.9: Statistische verwerking simulatie	NUTTIG
4: Gebruikersinterface	
Use case 4.1: GUI voor de simulatie	NUTTIG
Use case 4.2: GUI voor routeplanner	NUTTIG
Use case 4.3: GUI voor statistische gegevens	NUTTIG

1.1. Trams en stations inlezen

Prioriteit:
VERPLICHT

Doel:
Inlezen van het schema van het metro net. De verschillende stations, hoe die met elkaar verbonden zijn en de verschillende trams.

Preconditie:
Een ASCII bestand met daarop een beschrijving van de stations en trams. (Zie Appendix A voor meer informatie over het XML formaat)

Succesvol einde:
Het systeem bevat een spoorschema met de verschillende stations, en informatie over alle trams.

Stappen:
1. Open invoerbestand
2. WHILE Bestand nog niet volledig ingelezen
2.1. Herken het soort element (STATION, TRAM)
2.2. Lees verdere informatie voor het element
2.3. IF Verifieer geldige informatie
2.3.1. THEN Voeg element toe aan het virtuele metronet
2.3.1. ELSE Foutboodschap + positioneer op volgende element in het bestand
3. Verifieer de consistentie van het metronet (zie Appendix A)
4. Sluit invoerbestand

Uitzonderingen:
2.1. [Onherkenbaar element] Foutboodschap + positioneer op volgende element in het bestand ⇒ verdergaan vanaf stap 2
2.2. [Ongeldige informatie] Foutboodschap + positioneer op volgende element in het bestand ⇒ verdergaan vanaf stap 2
3. [Inconsistent metronet] Foutboodschap ⇒ verdergaan vanaf stap 4

Voorbeeld:
Een metronet met drie stations (A,B,C), een spoor (12) en een tram (12).

```
<METRONET>
    <STATION>
        <naam>A</naam>
        <volgende>B</volgende>
        <vorige>C</vorige>
        <spoorNr>12</spoorNr>
    </STATION>
    <STATION>
        <naam>B</naam>
        <volgende>C</volgende>
        <vorige>A</vorige>
        <spoorNr>12</spoorNr>
    </STATION>
    <STATION>
        <naam>C</naam>
        <volgende>A</volgende>
        <vorige>B</vorige>
        <spoorNr>12</spoorNr>
    </STATION>
    <TRAM>
        <lijnNr>12</lijnNr>
        <snelheid>60</snelheid>
        <beginStation>A</beginStation>
    </TRAM>
</METRONET>
```

1.2. Trams en stations met type inlezen

Prioriteit:
VERPLICHT

Doel:
Sinds 30 januari beschikt De Lijn over een nieuw model van trams: de Stadslijner. Deze komen bovenop de oudere Albatros en PCC-trams. Momenteel zitten we echter in een overgangsfase waar all types tegelijk in gebruik zijn. Om tot een meer realistische simulatie te komen moet het mogelijk zijn onderscheid te maken tussen de drie types.

De Albatros tram en de Stadslijner zijn aanzienlijk langer is dan de oude PCC-trams. Hierdoor kunnen deze twee types niet stoppen aan bovengrondse haltes, aangezien het autoverkeer dan gehinderd wordt. Daarom moet het ook mogelijk zijn onderscheid te maken tussen twee types stations: het ondergrondse "metrostation" en een bovengrondse "halte".

Merk op dat het `snelheid` attribuut niet langer ondersteund moet worden, aangezien deze kunnen worden afgeleid uit het type van de tram (Zie Appendix B).

Referenties:
`https://nl.wikipedia.org/wiki/Voertuigenpark_van_De_Lijn#Urbos_100`
`https://nl.wikipedia.org/wiki/Albatros_(tram)`
`https://nl.wikipedia.org/wiki/Presidents'_Conference_Committee-tram`

Uitbreiding:
Use case 1.1: Trams en stations inlezen

Stappen:
[2.2, tijdens] Hou rekening met extra attribuut `type`, negeer het `snelheid` attribuut.

Uitzonderingen:
Geen

Voorbeeld:
Voorbeeld van verschillende types. Opgepast: om het kort te houden is dit voorbeeld niet volledig: de stations van lijn 9 ontbreken. Er is ook geen voorbeeld van een tram van het "Albatros" type.

```
<METRONET>
    <STATION>
        <naam>A</naam>
        <type>Metrostation</type>
        <volgende>B</volgende>
        <vorige>C</vorige>
        <spoor>15</spoor>
    </STATION>
    <STATION>
        <naam>B</naam>
        <type>Halte</type>
        <volgende>C</volgende>
        <vorige>A</vorige>
        <spoor>15</spoor>
    </STATION>
    <STATION>
        <naam>C</naam>
        <type>Metrostation</type>
        <volgende>A</volgende>
        <vorige>B</vorige>
        <spoor>15</spoor>
    </STATION>
    <TRAM>
        <lijnNr>9</lijnNr>
        <type>PCC</type>
        <beginStation>B</beginStation>
    </TRAM>
    <TRAM>
        <lijnNr>15</lijnNr>
        <type>Stadslijner</type>
        <beginStation>C</beginStation>
    </TRAM>
</METRONET>
```

1.3. Trams met voertuignummer inlezen

Prioriteit:
BELANGRIJK

Doel:
Om tot een meer realistische simulatie te komen moet het mogelijk zijn dat meerdere trams op éénzelfde spoor voorkomen. Daarvoor moet elke tram voorzien zijn van een voertuignummer.

Uitbreiding:
Use case 1.1: Trams en stations inlezen

Stappen:
[2.2, tijdens] Hou rekening met extra attribuut `voertuigNr`.

Uitzonderingen:
Geen

Voorbeeld:
Twee trams op lijn 12:

```
<METRONET>
    ...
    <TRAM>
        <lijnNr>12</lijnNr>
        <type>PCC</type>
        <voertuigNr>1</voertuigNr>
        <beginStation>A</beginStation>
    </TRAM>
    <TRAM>
        <lijnNr>12</lijnNr>
        <type>Albatros</type>
        <voertuigNr>2</voertuigNr>
        <beginStation>B</beginStation>
    </TRAM>
    ...
</METRONET>
```

1.4 Stations met meerdere sporen inlezen

Prioriteit:
NUTTIG

Doel:
Om tot een meer realistische simulatie te komen moet het mogelijk zijn dat een station meerdere sporen kan bevatten.

Opgepast: Als je Use case 1.2: Trams en stations met type inlezen implementeert, moet je rekening houden met verschillende types stations. Enkel een metrostation kan meerdere sporen bevatten, dus een halte heeft maximaal één spoor.

Uitbreiding:
Use case 1.1: Trams en stations inlezen
Use case 1.2: Trams en stations met type inlezen

Stappen:
[2.2, tijdens] Hou rekening met andere structuur (meerdere sporen per station)

Uitzonderingen:
Geen

Voorbeeld:
Drie stations met elk twee sporen, waarbij spoor 21 dezelfde route doet als spoor 12 in de omgekeerde richting (zie volgende pagina). Merk op dat met deze uitbreiding het invoerformaat voor stations wijzigt.

```
<METRONET>
    <STATION>
        <naam>A</naam>
        <type>Metrostation</type>
        <SPOOR>
            <spoorNr>12</spoorNr>
            <volgende>B</volgende>
            <vorige>C</vorige>
        </SPOOR>
        <SPOOR>
            <spoorNr>21</spoorNr>
            <volgende>C</volgende>
            <vorige>B</vorige>
        </SPOOR>
    </STATION>
    <STATION>
        <naam>B</naam>
        <type>Metrostation</type>
        <SPOOR>
            <spoorNr>12</spoorNr>
            <volgende>C</volgende>
            <vorige>A</vorige>
        </SPOOR>
        <SPOOR>
            <spoorNr>21</spoorNr>
            <volgende>A</volgende>
            <vorige>C</vorige>
        </SPOOR>
    </STATION>
    <STATION>
        <naam>C</naam>
        <type>Metrostation</type>
        <SPOOR>
            <spoorNr>12</spoorNr>
            <volgende>A</volgende>
            <vorige>B</vorige>
        </SPOOR>
        <SPOOR>
            <spoorNr>21</spoorNr>
            <volgende>B</volgende>
            <vorige>A</vorige>
        </SPOOR>
    </STATION>
</METRONET>
```

2.1. Simpele uitvoer (oude versie)

Prioriteit:
VERPLICHT

Doel:
Uitvoer van alle informatie in de simulatie.

Preconditie:
Het systeem bevat een schema van het metronet.

Succesvol einde:
Het systeem heeft een tekstbestand (ASCII) aangemaakt, waarin de informatie over het virtuele metronet netjes is uitgeschreven.

Stappen:
1. Maak uitvoerbestand
2. WHILE Nog stations beschikbaar
2.1. Schrijf station-gegevens uit naar bestand
3. WHILE Nog trams beschikbaar
3.1. Schrijf tram-gegevens uit naar bestand
4. Sluit uitvoerbestand

Uitzonderingen:
Geen

Voorbeeld:
Weergave van drie stations, met vermelding van de volgende en voorgaande stations, alsook de weergave van een tram met vermelding van het huidige station.

```
Station A
<- Station C
-> Station B
Spoor 12

Station B
<- Station A
-> Station C
Spoor 12

Station C
<- Station B
-> Station A
Spoor 12

Tram 12 in Station A
```

2.2. Simpele uitvoer (herzien)

Prioriteit:
VERPLICHT

Doel:
Om een beter overzicht te krijgen van het metronet wil de klant een overzichtelijkere uitvoer van alle informatie. Zo moet bijvoorbeeld de volgorde van stations op één spoorlijn duidelijker naar voor komen.

Opgepast: afhankelijk van welke use-cases je implementeert, zal de uitvoer van deze use-case verschillen. Elke use-case kan namelijk extra informatie bijdragen. Indien bepaalde use-cases niet worden geïmplementeerd, mag de relevante output hiervoor worden weggelaten.

Preconditie:
Het systeem bevat een grondplan voor een virtueel metronet.

Succesvol einde:
Het systeem heeft een tekstbestand (ASCII) aangemaakt, waarin de informatie over het virtuele metronet netjes is uitgeschreven.

Uitbreiding:
Dit is een aanpassing van Use case 2.1: Simpele uitvoer (oude versie)

Stappen:
1. Maak uitvoerbestand
2. WHILE Nog stations beschikbaar
2.1. Schrijf stationgegevens uit naar bestand
3. WHILE nog trams op spoorlijn
3.1. Schrijf tramgegevens uit naar bestand
4. Sluit uitvoerbestand

Uitzonderingen:
Geen

Voorbeeld:
In dit voorbeeld zie je de output voor meerdere use cases. Op bepaalde plaatsen staat er "..." om herhaling te voorkomen, maar in de implementatie moet je hier concrete informatie printen, en dus niet letterlijk "..."

```
--== STATIONS ==--
= Station A =
* Spoor 12:
    -> Station B
    <- Station C

* Spoor 33:
...

= Station B =
* Spoor 12:
    -> Station C
    <- Station A

* ...

= Station C =
* Spoor 12:
    -> Station A
    <- Station B

* ...

--== TRAMS ==--
Tram 12 nr 1
    type: PCC
    snelheid: 40
    huidig station: A
    reparatiekosten: 62 euro

Tram 12 nr 353
...
```

2.3. Geavanceerde uitvoer

Prioriteit:
BELANGRIJK

Doel:
De toestand van het metronet wordt grafisch weergegeven.

Preconditie:
Het systeem is correct geïnitialiseerd.

Succesvol einde:
Het systeem heeft een tekstbestand (ASCII) uitgevoerd, waarin de toestand van het metronet staat beschreven.

Stappen:
1. Maak uitvoerbestand
2. Teken gegevens uit voor de toestand van het metronet
3. Sluit uitvoerbestand

Uitzonderingen:
Geen

Voorbeeld:
Er zijn vier stations (A, B, C, D), twee sporen en drie trams (T):

```
=A===B===C=
 T   T

=A===D===C=
         T
```

2.4. Grafische 3D rendering

Prioriteit:
BELANGRIJK

Doel:
Om het stadsbestuur van de stad Antwerpen te overtuigen had onze klant graag een 3D visualisatie gehad van het rondrijden van de trams. Hiervoor kan je een standaard interface voor je graphics engine gebruiken.

Preconditie:
Het systeem is correct geïnitialiseerd.

Succesvol einde:
Elke beweging van trams wordt weergegeven in een 3D omgeving.

3.1. Rijden van trams

Prioriteit:
VERPLICHT

Doel:
Simuleren van rondrijdende trams op het metronet.

Preconditie:
Het systeem bevat een grondplan voor een virtueel metronet.

Succesvol einde:
Een tram bevindt zich op een nieuwe locatie in het metronet. Het systeem heeft een boodschap afgedrukt met de details van de verplaatsing.

Stappen:
1. Voer verplaatsing uit voor tram op gegeven spoor in gegeven station
2. Schrijf overzicht uit

Uitzonderingen:
Geen

Voorbeeld:
Stel dat tram tijdens deze stap van de simulatie verplaatste van station A naar station B:

```
Tram 12 reed van station A naar station B.
```

3.2. Automatische simulatie (oude versie)

Prioriteit:
VERPLICHT

Doel:
Simulatie automatisch laten lopen voor een gegeven tijd.

Preconditie:
Het systeem bevat een grondplan voor een virtueel metronet.
Use case 3.1: Rijden van trams is geïmplementeerd.

Succesvol einde:
De simulatie stopt na het gegeven aantal stappen.

Stappen:
1. WHILE huidige tijd < eind tijd
1.1 Use case 3.1: Rijden van trams uit

3.3. Automatische simulatie (herzien)

Prioriteit:
VERPLICHT

Doel:
Simulatie automatisch laten lopen voor een gegeven tijd.

Preconditie:
Het systeem bevat een grondplan voor een virtueel metronet.
Use case 3.5: Rijden van meerdere trams is geïmplementeerd.

Succesvol einde:
De simulatie stopt na het gegeven aantal stappen.

Uitbreiding:
Dit is een aanpassing van Use case 3.2: Automatische simulatie (oude versie)

Stappen:
1. WHILE huidige tijd < eind tijd
1.1 Use case 3.5: Rijden van meerdere trams uit

3.4. Rijden van trams met type

Prioriteit:
VERPLICHT

Doel:
Een specifieke tram zal zich anders gedragen afhangend van zijn type, zoals bijvoorbeeld het feit dat een Albatros-tram of Stadslijner niet kan stoppen in een bovengrondse “halte”. Zie Appendix B voor meer informatie over de verschillende soorten trams. Hiervoor moet je eerst Use case 1.2: Trams en stations met type inlezen implementeren.

Uitbreiding:
Use case 3.1: Rijden van trams
Use case 3.5: Rijden van meerdere trams

Uitzonderingen:
Geen

3.5. Rijden van meerdere trams

Prioriteit:
VERPLICHT

Doel:
Om tot een meer realistische simulatie te komen moeten meerdere trams tegelijkertijd kunnen rondrijden op hetzelfde spoor. Hiervoor moet je ook Use case 3.1: Rijden van trams implementeren.

Preconditie:
Het systeem bevat een grondplan voor een virtueel metronet.
Use case 3.1: Rijden van trams is geïmplementeerd.

Succesvol einde:
Alle trams bevinden zich op een nieuwe locatie in het metronet, het systeem heeft een boodschap afgedrukt met de details over de verplaatsingen.

Uitbreiding:
Use case 3.1: Rijden van trams

Stappen:
1. FOR EACH tram
1.1 voer Use case 3.1: Rijden van trams uit

3.6. Botspreventie

Prioriteit:
BELANGRIJK

Doel:
Om tot een meer realistische simulatie te komen moeten meerdere trams tegelijkertijd kunnen rondrijden op hetzelfde spoor. Hiervoor moet je ook Use case 3.5: Rijden van meerdere trams implementeren. Trams op eenzelfde spoor kunnen elkaar niet inhalen, maar moeten wachten tot de tram voor hen in een bepaald station vertrekt.

Preconditie:
Het systeem bevat een grondplan voor een virtueel metronet.

Succesvol einde:
Een tram die naar station X moet, moet wachten in het vorige station als er nog een tram in station X staat.

Uitbreiding:
Use case 3.5: Rijden van meerdere trams

3.7. Defecten en reparaties

Prioriteit:
VERPLICHT

Doel:
De PCC-trams dateren van de jaren 60 en 70 en zijn dus verouderd, met defecten en motorpech als gevolg. De Lijn wil met behulp van deze simulatietool in kaart brengen wat hiervan de impact is op het tramverkeer.

In de XML input files zal een PCC tram twee extra attributes hebben genaamd `aantalDefecten` en `reparatieTijd`. Het aantal defecten bepaalt om de hoeveel stations de tram in panne zal vallen. De reparatietijd bepaalt hoeveel stappen van de simulatie de tram moet wachten vooraleer de tram terug kan vertrekken.

Preconditie:
Het systeem bevat een grondplan voor een virtueel metronet.

Succesvol einde:
De tram zal tijdens het rijden om de zoveel stations in panne vallen. Zodra een tram in panne valt, zal de tram een aantal stappen wachten vooraleer de tram terug vertrekt.

Uitbreiding:
Use case 1.1: Trams en stations inlezen
Use case 3.3: Automatische simulatie (herzien)

Stappen voor Use case 1.1: Trams en stations inlezen
[2.2, tijdens] Hou rekening met extra attribuut `reparatieKost`.

Stappen voor Use case 3.3: Automatische simulatie (herzien)
[1.1, tijdens] Hou rekening met het feit dat een tram in panne kan vallen, defect is, of terug gerepareerd is.

3.8 Reparatiekosten

Prioriteit:
NUTTIG

Doel:
Bovenop de impact van defecten en motorpech op het tramverkeer, wilt De Lijn ook kunnen inschatten wat de kost is van reparaties. Telkens dat een PCC-tram gerepareerd moet worden, zal de totale reparatiekost verhoogd worden.

In de XML input files zal een PCC tram een `reparatieKost` attribuut hebben. De reparatiekost is het bedrag dat bij de totale kost zal worden opgeteld, telkens dat een tram in panne valt. De totale reparatiekost moet per tramtoestel apart berekend worden.

Preconditie:
Het systeem bevat een grondplan voor een virtueel metronet.

Succesvol einde:
Wanneer een PCC tram in panne valt, zal de reparatie geld kosten en zal het totaal aantal kosten aangepast zijn met deze reparatiekost.

Uitbreiding:
Use case 1.1: Trams en stations inlezen
Use case 2.2: Simpele uitvoer (herzien)
Use case 3.3: Automatische simulatie (herzien)

Stappen voor Use case 1.1: Trams en stations inlezen
[2.2, tijdens] Hou rekening met extra attribuut `reparatieKost`.

Stappen voor Use case 2.2: Simpele uitvoer (herzien)
[3.1, tijdens] Hou rekening met trams die defect zijn, de reparatiekost, en of de tram al (gedeeltelijk) gerepareerd is.

Stappen voor Use case 3.3: Automatische simulatie (herzien)
[1.1, tijdens] Als een tram in panne valt, tel de reparatiekost op bij de totale kost.

3.9 Statistische verwerking simulatie

Prioriteit:
NUTTIG

Doel:
Gedurende de uitvoering worden er gegevens verzameld over het metronet, zoals de reparatiekost per tram, totale reparatiekost voor het metronet, hoeveel trams er voorbij een station zijn gereden, etc.

Preconditie:
Het systeem bevat een grondplan voor een virtueel metronet.

Succesvol einde:
Rapport afgedrukt met statistische gegevens.

Uitbreiding:
Use case 3.3: Automatische simulatie (herzien)

Stappen:
1. WHILE simulatie actief
1.1 Verzamel gegevens
2. Bereken waarden
3. Druk rapport af

4.1. GUI voor de simulatie

Prioriteit:
NUTTIG

Doel:
Een gebruiksvriendelijke gebruikersinterface hebben voor het controleren van de simulatie. Deze bevat knoppen voor onder andere start en stop, alsook volgende en vorige stap.

Preconditie:
Het systeem bevat een grondplan voor een virtueel metronet.

Succesvol einde:
De simulatie kan bestuurd worden aan de hand van een grafische gebruikersinterface.

4.2. GUI voor de routeplanner

Prioriteit:
NUTTIG

Doel:
Een gebruiksvriendelijke interface hebben voor het plannen van routes binnen het metronet, met inbegrip van stations waar men moet overstappen.

Preconditie:
Het systeem bevat een grondplan voor een virtueel metronet.

Succesvol einde:
De gebruiker ziet de route die genomen moet worden om het eindstation te bereiken. De gebruiker ziet welke tramlijnen genomen moeten worden en in welke stations er een overstap gemaakt moet worden.

Stappen:
1. Duid het beginstation aan in de GUI
2. Duid het eindstation aan in de GUI
3. Een route tussen deze twee stations wordt weergegeven in de GUI

Uitzonderingen:
3. [Er bestaat geen route tussen twee stations] Geef een foutboodschap weer met vermelding van de namen van de stations en het feit dat er geen route bestaat tussen deze twee stations. Vraag de gebruiker om andere stations aan te duiden.

4.3. GUI voor statistische gegevens

Prioriteit:
NUTTIG

Doel:
Visualizatie van de statistische gegevens verzameld in Use case 3.9: Statistische verwerking simulatie.

Preconditie:
Het systeem heeft statistische gegevens verzameld.

Succesvol einde:
De statistische gegevens worden weergegeven in een grafiek door middel van een grafische gebruikersinterface (GUI).

Uitbreiding:
Use case 3.9: Statistische verwerking simulatie

A Invoer formaat

Het invoerformaat voor het virtueel metronet is zodanig gekozen dat nieuwe attributen en elementen makkelijk kunnen worden toegevoegd.

```
File = "<METRONET>" ElementLijst "</METRONET>"
ElementLijst = Element { Element ... }
Element = "<" ElementType ">" AttribuutLijst "</" ElementType ">"
ElementType = "STATION" | "TRAM" | "SPOOR"
AttribuutLijst = Attribuut { Attribuut ... }
Attribuut = "<" AttribuutType ">" AttribuutWaarde "</" AttribuutType ">"
AttribuutType = (zie tabel)
AttribuutWaarde = Primitief
Primitief = Integer | String
Integer = Digit { Digit ... }
Digit = "0" | ... | "9"
String = Letter { Letter ... }
Letter = "a" | ... | "z" | "A" | ... | "Z"
```

Merk op dat de attribuutlijst een relatief vrij formaat heeft wat sterk zal afhangen van het soort element dat gedefinieerd wordt. De volgende tabel toont de attributen voor elk element:

Element type	Attribuut types (verplicht)	Attribuut types (optioneel)
STATION	naam, volgende, vorige, spoorNr, type	/
TRAM	lijnNr, beginStation, type, **snelheid***	voertuigNr
SPOOR	/	spoorNr, volgende, vorige

*** Enkel wanneer je Use case 1.2: Trams en stations met type inlezen nog niet hebt geïmplementeerd.**

Bovendien zal afhankelijk van het attribuuttype slechts een bepaalde attribuutwaarde toegelaten zijn:

Attribuut	Waarde
naam, vorige, volgende, beginStation, type	String
spoorNr, lijnNr, voertuigNr, reparatiekost, **snelheid***	Integer

*** Enkel wanneer je Use case 1.2: Trams en stations met type inlezen nog niet hebt geïmplementeerd.**

Bovendien moet de openings tag steeds overeenkomen met de sluitingstag. Vandaar dat tijdens de invoer moet gecontroleerd worden of de invoer al dan niet geldig is.

Het bestand met het in te lezen metronet wordt met de hand geschreven. Om het ingelezen metronet te kunnen simuleren moet de informatie consistent zijn.

Een metronet is consistent als:

- elk station is verbonden met een voorgaand en een volgend station voor elk spoor.
- elke tram heeft een lijn die overeenkomt met een spoor in zijn beginstation.
- er geen sporen zijn waarvoor geen tram bestaat.
- het startstation van een tram is een geldig station in het metronet.
- elk spoor komt maximaal 1 keer voor in een station.
- er zijn geen twee trams met hetzelfde voertuignummer

B Tram data

De volgende tabel toont de data voor elk type tram:

	PCC	Albatros	Stadslijner
Snelheid	40	70	70
Bedient	metrostation, halte	metrostation	metrostation