Metro Simulatie

Documentsoort:	Behoeftespecificatie
Versie:	1.0
Datum:	27 februari 2020
Auteurs:	Brent van Bladel
Status:	In development

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 1 februari 2017 is de stad Antwerpen in de kernstad en op linkeroever een lage-emissiezone. In deze lage-emissiezone mogen enkel voertuigen rijden die aan bepaalde milieucriteria voldoen. Op 1 januari 2020 zijn deze criteria aangescherpt. Aangezien een heleboel voertuigen daarom niet langer de stad in mogen, verwacht de gewestelijke vervoersmaatschappij De Lijn een toename in het aantal passagiers. Het is voor De Lijn van groot belang op voorhand een duidelijk beeld te hebben van de situatie (qua bezetting en piekmomenten) van het metronet. Daarom heeft De Lijn geopteerd een simulatie model te laten ontwikkelen dat het tramverkeer kan simuleren.

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

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 van die use cases geïmplementeerd worden. Een typische use-case bevat de volgende onderdelen:

- **Refertenummer & titel:**
  Wordt gebruikt om naar een bepaalde use-case te verwijzen.
- **Prioriteit:**
  De specificatie van een systeem vraagt meer dan wat binnen de voorziene tijd op te leveren is. 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 uitgangspunten bij aanvang van de use-case.
- **Succesvol einde:**
  Summiere beschrijving van wat opgeleverd zal worden 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 (a) verwijst naar het nummer van de stap waar het probleem kan optreden, (b) bevat een conditie die aangeeft wanneer het probleemgeval optreedt, (c) omschrijft heel kort (een lijn) hoe het probleem behandeld zal worden.
- **Voorbeeld:**
  Een voorbeeld van wat in- of uitgevoerd kan worden.

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

- **Uitbreiding:**
  Een referte 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 (a) verwijst naar het nummer van de stap die uitgebreid wordt, (b) zegt of de uitbreiding voor, na of tijdens de normale stap zal gebeuren, (c) omschrijft wat precies in de uitbreiding zal gebeuren.

4 Overzicht

Use-Case	**Prioriteit**
1: Invoer	
1.1. Trams en stations inlezen	VERPLICHT
2: Uitvoer	
2.1. Simpele uitvoer	VERPLICHT
3: Simulatie	
3.1. Rijden van trams	VERPLICHT
3.2. Automatische simulatie	BELANGRIJK

1.1. Trams en stations inlezen

Prioriteit:
VERPLICHT

Doel:
Inlezen van het schema van het metronet. 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 niet 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 de simulatie
2.3.1. ELSE Foutboodschap + positioneer op volgende element in het bestand
3. Verifieer consistentie van de metronet
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. [Inconsistente metronet] Foutboodschap ⇒ verdergaan vanaf stap 4

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

```
<STATION>
    <naam>A</naam>
    <volgende>B</volgende>
    <vorige>C</vorige>
    <spoor>12</spoor>
</STATION>
<STATION>
    <naam>B</naam>
    <volgende>C</volgende>
    <vorige>A</vorige>
    <spoor>12</spoor>
</STATION>
<STATION>
    <naam>C</naam>
    <volgende>A</volgende>
    <vorige>B</vorige>
    <spoor>12</spoor>
</STATION>
<TRAM>
    <lijn>12</lijn>
    <zitplaatsen>32</zitplaatsen>
    <snelheid>60</snelheid>
    <beginStation>A</beginStation>
</TRAM>
```

2.1. Simpele uitvoer

Prioriteit:
VERPLICHT

Doel:
Uitvoer van alle informatie in de simulatie.

Preconditie:
Het systeem bevat een schema van de virtuele metronet.

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

Stappen:
1. Open uitvoerbestand
2. WHILE Nog banen beschikbaar
2.1. Schrijf baan-gegevens uit
3. WHILE Nog voertuigen beschikbaar
3.1. Schrijf voertuig-gegevens uit
4. Sluit uitvoerbestand

Uitzonderingen:
Geen

Voorbeeld:
Gegeven de input van 1.1

```
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, 32 zitplaatsen
```

3.1. Rijden van trams

Prioriteit:
VERPLICHT

Doel:
Simuleren van een rondrijdende tram in 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:
Gegeven de input van 1.1

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

3.2. Automatische simulatie

Prioriteit:
BELANGRIJK

Doel:
Simulatie automatisch laten lopen voor een gegeven tijd.

Preconditie:
Het systeem bevat een schema van de virtuele metronet.

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

Stappen:
1. WHILE huidige tijd < eind tijd
1.1 FOR elke tram in het metronet
1.1.1 voer use case 3.1 uit op de tram

A Invoer formaat

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

```
MetroNet = { Element }
Element = "<" ElementType ">" AttribuutLijst "</" ElementType ">"
ElementType = "STATION" | "TRAM"
AttribuutLijst = Attribuut { Attribuut }
Attribuut = "<" AttribuutType ">" AttribuutWaarde "</" AttribuutType ">"
AttribuutType = "naam" | "vorige" | "volgende" | "spoor" | "lijn"
                       | "zitplaatsen" | "snelheid" | "beginStation"
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	Attribuut (verplicht)
Station	naam, volgende, vorige, spoor
Tram	lijn, zitplaatsen, snelheid, beginStation

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

Attribuut	Waarde
naam, vorige, volgende, beginStation	String
spoor, lijn, zitplaatsen, snelheid	Integer

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 heeft exact 1 spoor.
- elk station is verbonden met een voorgaand en een volgend station met hetzelfde spoor.
- elke tram heeft een lijn die overeenkomt met een spoor in zijn beginstation.
- elk spoor heeft exact 1 tram.
- het startstation van een tram is een geldig station in het metronet.
- elk spoor komt maximaal 1 keer voor in een station.