Metro Simulatie

Documentsoort:	Behoeftespecificatie
Versie:	1.0
Datum:	2 maart 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
2: Uitvoer	
Use case 2.1: Simpele uitvoer	VERPLICHT
3: Simulatie	
Use case 3.1: Rijden van trams	VERPLICHT
Use case 3.2: Automatische simulatie	BELANGRIJK

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).

```
<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>
```

2.1. Simpele uitvoer

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
```

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

Prioriteit:
BELANGRIJK

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

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 = (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
STATION	naam, volgende, vorige, spoorNr
TRAM	lijnNr, beginStation, snelheid

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

Attribuut	Waarde
naam, vorige, volgende, beginStation	String
spoorNr, lijnNr, 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 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