Subway Simulation

Document:	System Specification
Version:	1.0
Date:	16 October 2019
Author:	Brent van Bladel
Status:	Delivered

1 Summary

This document contains the specification for a computersystem that can execute a subway simulation. It has been written for the course "Project Software Engineering" (1st bachelor computer science - University of Antwerpen).

2 Context

Since February 1st 2017, the citycenter of Antwerp has become a low emission zone. Only vehicles that meet strict environmental regulation may enter this zone. Since a lot of vehicles are no longer allowed in the city center, regional transport company De Lijn expects an increase in its number of passengers. It is important for them to be able to predict subway traffic in advance (in terms of occupation and peak times). Therefore, De Lijn has opted to develop a simulation model of subway traffic.

The University of Antwerp has been asked to develop this simulation. During the first year of the computer science bachelor, students will be tasked to work on this project in the "Computer Graphics" and "Project Software Engineering" courses.

3 Legend

The requirements specification has been drafted on the basis of use-cases. Each use-case describes a small part of the desired functionality. The intention is that during each phase of the project different use-cases are implemented. A typical use-case contains the following components:

- **Reference number & title:**
  Used to identify or refer to a use-case.
- **Priority:**
  The specification of the system demands more than what can be delivered within the foreseen time. That is why we use the priority of a use-case to indicate to what extent its functionality is important. The priority can be (in order of importance): REQUIRED (this use case must be completed), IMPORTANT (not essential but preferably deliver), USEFUL (interesting but can be omitted).
- **Goal:**
  Brief description of the goal of the use case, i.e. what the use case contributes to the entire functionality.
- **Precondition:**
  Brief description of the required properties at the start of the use-case.
- **Postcondition:**
  Brief description of the required properties at the end of the use-case.
- **Steps:**
  A sequential description of how the use-case executes if everything goes well (the so-called "happy day scenario "). The steps are numbered and may include control instructions (WHILE, IF, ...).
- **Exceptions:**
  A list of possible deviations from the happy day scenario and how they should be treated. An exception (a) refers to the number of the step where the exception may occur, (b) contains a condition that indicates when the exception occurs, and (c) describes very briefly how the exception will be treated.
- **Example:**
  An example of the input or output.

Sometimes a use-case is an extension of another use-case. Then the following components are relevant:

- **Extension:**
  A reference to the use-case that is being extended.

- **Steps:**
  A list of additional and / or modified steps with regard to the use-case that is being extended.
  An extension (a) refers to the step number being extended, (b) states whether the extension is before, after, or during the step, and (c) describes what exactly will happen in the extension.

4 Overview

Use-Case	Priority
1: Input	
1.1 Parsing of trams and stations	REQUIRED
2: Output	
2.1 Simple ouput	REQUIRED
3: Simulation	
3.1 Moving trams	REQUIRED
3.2 Automatic simulation	IMPORTANT

1.1. Parsing of trams and stations

Priority:
REQUIRED

Goal:
Parsing the schedule of the subway network: the different stations, how they are connected to each other, and the different trams.

Precondition:
An ASCII file with a description of the stations and trams. (See Appendix A for more information about the XML format)

Postcondition:
The system contains a virtual subway network with the different stations and data on all trams.

Steps:
1. Open inputfile
2. WHILE Not at end of file
2.1. Detect the type of element (STATION, TRAM)
2.2. Read data of the element
2.3. IF Verify data is valid
2.3.1. THEN Add the element to the virtual subway network
2.3.1. ELSE Errormessage + go to next element in the file
3. Verify consistency of the subway network
4. Close inputfile

Exceptions:
2.1. [Unrecognized element] Errormessage + go to next element in the file ⇒ continue from step 2
2.2. [Invalid data] Errormessage + go to next element in the file ⇒ continue from step 2
3. [Inconsistent subway network] Errormessage ⇒ continue from step 4

Example:
A subway network with three stations (A,B,C), one track (12) and one tram (12).

```
<STATION>
    <name>A</name>
    <next>B</next>
    <previous>C</previous>
    <track>12</track>
</STATION>
<STATION>
    <name>B</name>
    <next>C</next>
    <previous>A</previous>
    <track>12</track>
</STATION>
<STATION>
    <name>C</name>
    <next>A</next>
    <previous>B</previous>
    <track>12</track>
</STATION>
<TRAM>
    <line>12</line>
    <capacity>32</capacity>
    <speed>60</speed>
    <startStation>A</startStation>
</TRAM>
```

2.1. Simple output

Priority:
REQUIRED

GOAL:
Output all data from the virtual subway system.

Precondition:
The system contains a virtual subway network with the different stations and data on all trams.

Postcondition:
The system generated an ASCII file that contains all data from the virtual subway network.

Steps:
1. Open outputfile
2. WHILE Stations available
2.1. Write data of station
2.2. Write data per track, i.e. capacity of trams on the track
3. Close outputfile

Exceptions:
None

Example:
Given the input from 1.1

```
Station A
<- Station C
-> Station B
Track 12: Tram with 32 seats

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

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

3.1. Moving trams

Priority:
REQUIRED

Goal:
Simulation of trams on the subway network.

Precondition:
The system contains a virtual subway network with the different stations and data on all trams.

Postcondition:
A tram is located on a new location in the subway network. The system has printed a message with the details of the movement.

Steps:
1. Carry out movement for tram on given track in given station
2. Write overview

Exceptions:
None

Example:
Given the input from 1.1

```
Tram 12 moved from station A to station B.
```

3.2. Automatic simulation

Priority:
IMPORTANT

Goal:
Run the simulation automatically for a given amount of time.

Precondition:
The system contains a virtual subway network with the different stations and data on all trams.

Postcondition:
The simulation has halted after the given amount of time

Steps:
1. WHILE Current time < end time
1.1 Execute use-case 3.1
1.2 Increase current time by one

A Input format

The input format for the virtual subway network has been chosen in such a way that new attributes and elements can easily be added.

```
SubwaySystem = { Element }
Element = "<" ElementType ">" AttributeList "</" ElementType ">"
ElementType = "STATION" | "TRAM"
AttributeList = Attribute { Attribute }
Attribute = "<" AttributeType ">" AttributeValue "</" AttributeType ">"
AttributeType = "name" | "previous" | "next" | "track" | "line" | "capacity"
| "speed" | "startStation"
AttributeValue = Primitive
Primitive = Integer | String
Integer = Digit { Digit }
Digit = "0" ... "9"
String = Character { Character }
Character = "a" ... "z" | "A" ... "Z"
```

Note that the AttributeList has a relatively free format which will strongly depend on the type of element defined. The following table shows the attributes for each element:

ElementType	Attribute
STATION	name, previous, next, track
TRAM	line, capacity, startStation, speed

In addition, depending on the AttributeType, only one specific AttributeValue is allowed:

AttributeType	AttributeValue
name, previous, next, startStation	String
track, line, capacity, speed	Integer

In addition, the opening tag must always correspond to the closing tag. This is why it is necessary to check whether or not the input is valid during parsing.

The inputfile containing the subway network is written by hand. In order to simulate the subway system, the information must be consistent.

The subway system is consistent if:

- each station has exactly one track
- each station is connected to a previous station and to a next station that contains the same track
- each tram has a line that corresponds to a track in its starting station
- each track contains exactly one tram
- the starting station of a tram is a valid station in the subway network
- each track occurs at most once in every station