Metro Simulatie

Document:	System Specification
Version:	2.0
Date:	20 November 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 Overzicht

Use-Case	**Prioritity**
1: Input	
1.1 Parsing of trams and stations	REQUIRED
1.2 Parsing of trams and stations with type	REQUIRED
1.3 Parsing of trams with vehicle number	IMPORTANT
1.4 Parsing of stations with multiple tracks	USEFUL
2: Output	
2.2 Simple ouput	REQUIRED
2.2 Grafical representation	IMPORTANT
2.3 Integration with Graphics	USEFUL
3: Simulation	
3.1 Moving trams	REQUIRED
3.2 Simulation with multiple trams	REQUIRED
3.3 Automatic simulation	REQUIRED
3.4 Advanced time simulation	IMPORTANT
3.5 Collision prevention	IMPORTANT
3.6 Passenger simulation	USEFUL
3.7 Turnover per tram	USEFUL
3.8 Statistical data processing	USEFUL
4: Userinterface	
4.1 GUI for simulation	USEFUL
4.2 GUI for moving trams	USEFUL
4.3 GUI for statistical data	USEFUL

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>
```

1.2. Parsing of trams and stations with type

Prioritity:
REQUIRED

Goal:
In May 2015, DeLijn upgraded part of its trams to a new model: the Albatross. The goal is to eventually replace all the old PCC trams, yet we are currently still in a transition phase where both types of tram are being used at the same time. To achieve a more realistic simulation, it must be possible to distinguish between the two types of tram. The albatross tram has a length of 42 meters, which is considerably longer than the old PCC trams. As a result, they cannot stop at above-ground stops, since other traffic would be hindered. It must therefore also be possible to distinguish between two types of stations: the underground ”metro station” and an above-ground ”stop”. Note that the capacity and speed attributes are no longer necessary, as they can be deduced from the tram type (see Appendix B).

Extension:
Use Case 1.1

Steps:
[2, during] Parse the extra ’type’ attribute

Exceptions:
None

Example:
An example of each type. Note that this example is not a complete input.

```
<STATION>
    <name>A</name>
    <next>B</next>
    <previous>C</previous>
    <track>7</track>
    <type>stop</type>
</STATION>
<STATION>
    <name>D</name>
    <next>E</next>
    <previous>F</previous>
    <track>15</track>
    <type>station</type>
</STATION>
<TRAM>
    <line>7</line>
    <type>PCC</type>
    <startStation>A</startStation>
</TRAM>
<TRAM>
    <line>15</line>
    <type>Albatross</type>
    <startStation>D</startStation>
</TRAM>
```

1.3. Parsing of trams with vehicle number

Prioritity:
IMPORTANT

Goal:
To achieve a more realistic simulation, it should be possible for multiple trams to ride the same line. A vehicle number is then required to identify trams.

Extension:
Use Case 1.1

Steps:
[2, during] Parse the extra 'vehicle number' attribute

Exceptions:
None

Example:
Two trams on line 12, given the input from 1.1

```
<TRAM>
    <line>12</line>
    <vehicle>1</vehicle>
    <type>PCC</type>
    <startStation>A</startStation>
</TRAM>
<TRAM>
    <line>12</line>
    <vehicle>2</vehicle>
    <type>PCC</type>
    <startStation>B</startStation>
</TRAM>
```

1.4 Parsing of stations with multiple tracks

Prioritity:
USEFUL

Goal:
To achieve a more realistic simulation, it should be possible for a station to contain multiple tracks.

Extension:
Use Case 1.1

Steps:
[2, during] Parse the extra 'track' element and its attributes

Exceptions:
None

Example:
Three stations with two tracks each, where track 21 follows the same route as track 12 in the other direction. Note that, with this extension of use case 1.1, the structure of the input XML is changed.

```
<STATION>
    <name>A</name>
    <TRACK>
        <track>12</track>
        <next>B</next>
        <previous>C</previous>
    </TRACK>
    <TRACK>
        <track>21</track>
        <next>C</next>
        <previous>B</previous>
    </TRACK>
</STATION>
<STATION>
    <name>B</name>
    <TRACK>
        <track>12</track>
        <next>C</next>
        <previous>A</previous>
    </TRACK>
    <TRACK>
        <track>21</track>
        <next>A</next>
        <previous>C</previous>
    </TRACK>
</STATION>
<STATION>
    <name>C</name>
    <TRACK>
        <track>12</track>
        <next>A</next>
        <previous>B</previous>
    </TRACK>
    <TRACK>
        <track>21</track>
        <next>B</next>
        <previous>A</previous>
    </TRACK>
</STATION>
```

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
```

2.2. Grafical representation

Prioritity:
IMPORTANT

Goal:
The state of the subway system is displayed graphically.

Precondition:
The system has been initialized correctly.

Postcondition:
The system has generated a text file (ASCII) in which the state of the subway system is described.

Steps:
1. Open the outputfile
2. Draw the current state of the subway system
3. Close the outputfile

Exceptions:
None
Voorbeeld:
In case there are three STATIONS (A, B, C) and two trams (T):

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

2.3 Integration with Graphics

Prioritity:
USEFUL

Goal:
To convince the city authorities of Antwerp, our client would like to have a 3D visualization of the simulation. For this you can use a graphics engine of choice.

Precondition:
The system has been initialized correctly.

Postcondition:
Every movement of trams is displayed in a 3D environment.

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. Simulation with multiple trams

Priority:
REQUIRED

Goal:
Run the simulation for multiple trams.

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

Postcondition:
The simulation has moved each tram to its next location in the subway network.

Steps:
1. FOREACH Tram
1.1 Execute use-case 3.1

3.3. Automatic simulation

Priority:
REQUIRED

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 3.2
1.2 Increase current time by one
```

3.4. Advanced time simulation

Prioritity:
IMPORTANT

Goal:
To achieve a more realistic simulation of trams, time should be simulated correctly. Start the simulation at 12:00:00, and run the simulation according to the timings in Appendix C.

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.

Extension:
Use case 3.3

3.5. Collision prevention

Prioritity:
IMPORTANT

Goal:
Trams on the same track cannot overtake each other, but must wait until the previous tram leaves the next station.

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

Postcondition:
A tram that has to go to station X waited in station X-1 if another tram already occupied station X.

Extension:
Use case 3.2

Steps:
[1.1, before]
1.1 IF next station is occupied
1.1.1 wait in current station

3.6. Passenger simulation

Prioritity:
USEFUL

Goal:
To achieve a more realistic simulation, every tram needs to keep track of the amount of passengers currently riding the tram.

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

Postcondition: The number of passengers on a tram can be given at any time during the simulation.

Extension:
Use case 3.1

Steps:
[1, before] 1.1 WHEN a tram leaves a station
1.1.1 a random number is generated between the current and maximum capacity
1.1.2 the tram's current capacity is increased by this number

[1, after] 1.2 WHEN a tram arrives at a station
1.2.1 a random number is generated between zero and the current capacity of the tram
1.2.2 the tram's current capacity is decreased by this number

3.7. Turnover per tram

Prioritity:
USEFUL

Goal:
To simulate the potential profit, every tram needs to keep track of its turnover.

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

Postcondition: The current turnover of a tram can be given at any time during the simulation.

Extension:
Use case 3.6

Steps:
[1.1.2, after] 1.1.3 the tram's current turnover is increased by 2 euro for each boarding passenger

3.8. Statistical data processing

Prioritity:
USEFUL

Goal:
During the simulation, relative data is collected regarding current capacity and turnover.

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

Postcondition:
Report generated with statistical data.

Extension:
Use case 3.6
Use case 3.7

Steps:
1. WHILE Simulation is running
1.1 Collect data from each tram
2. Write data to CSV file

4.1. GUI for simulation

Prioritity:
USEFUL

Goal:
Have a user interface for controlling the simulation. This includes a pause, play, next step, and previous step button.

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

Postcondition:
The simulation can be controlled manually using on a graphical user interface.

4.2. GUI for moving trams

Prioritity:
USEFUL

Goal:
Have a user interface for controlling trams individually.

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

Postcondition:
Trams can be controlled manually using on a graphical user interface.

Steps:
1. Choose a tram in the UI
2. Have the option to make the tram wait in, or immediately leave, the current station

4.3 GUI for statistical data

Prioritity:
USEFUL

Goal:
Visualization of the statistical data collected in use case 3.7.

Precondition:
The system has collected the statistical data.

Postcondition:
The statistical data is displayed in a graph using a graphical user interface.

Extension:
Use case 3.7

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" | "TRACK"
AttributeList = Attribute { Attribute }
Attribute = "<" AttributeType ">" AttributeValue "</" AttributeType ">"
AttributeType = "name" | "previous" | "next" | "track" | "line" | "type"
| "vehicle" | "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 possible attributes for each element:

ElementType	Attribute
STATION	name, previous, next, track, type
TRAM	line, startStation, type, vehicle
TRACK	previous, next, track

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

AttributeType	AttributeValue
name, previous, next, startStation, type	String
track, line, vehicle	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 at least one track
- each track 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
- the starting station of a tram is a valid station in the subway network
- each track occurs at most once in every station

B Tram data

The following table provides an overview of the data for each tram type.

	PCC	Albatross
Capacity	16	72
Speed	40	70
Stops at	stations, stops	stations

C Advanced time simulation

The time (in seconds) required for a tram to move from one station to the next can be calculated using the following formula:

$$t = 3600 * \frac{distance}{speed}$$

However, since the tram won't always be travelling at its maximum speed, using this formula would result in unrealistically fast trams. We will therefore approximate a more realistic travelling time by doubling the result of the formula. Moreover, we can estimate that the distance between subway stations in Antwerp is about 1 kilometer. This results in

$$t = 2 * \left(3600 * \frac{1}{speed}\right)$$

or

$$t = \frac{7200}{speed}$$

For the time required to stand at a station, letting passengers on and off, we will use a constant of 60 seconds. Note that the Albatross trams will not take in passengers when the current station is an above-ground stop. In this case, the tram will leave after 1 second.