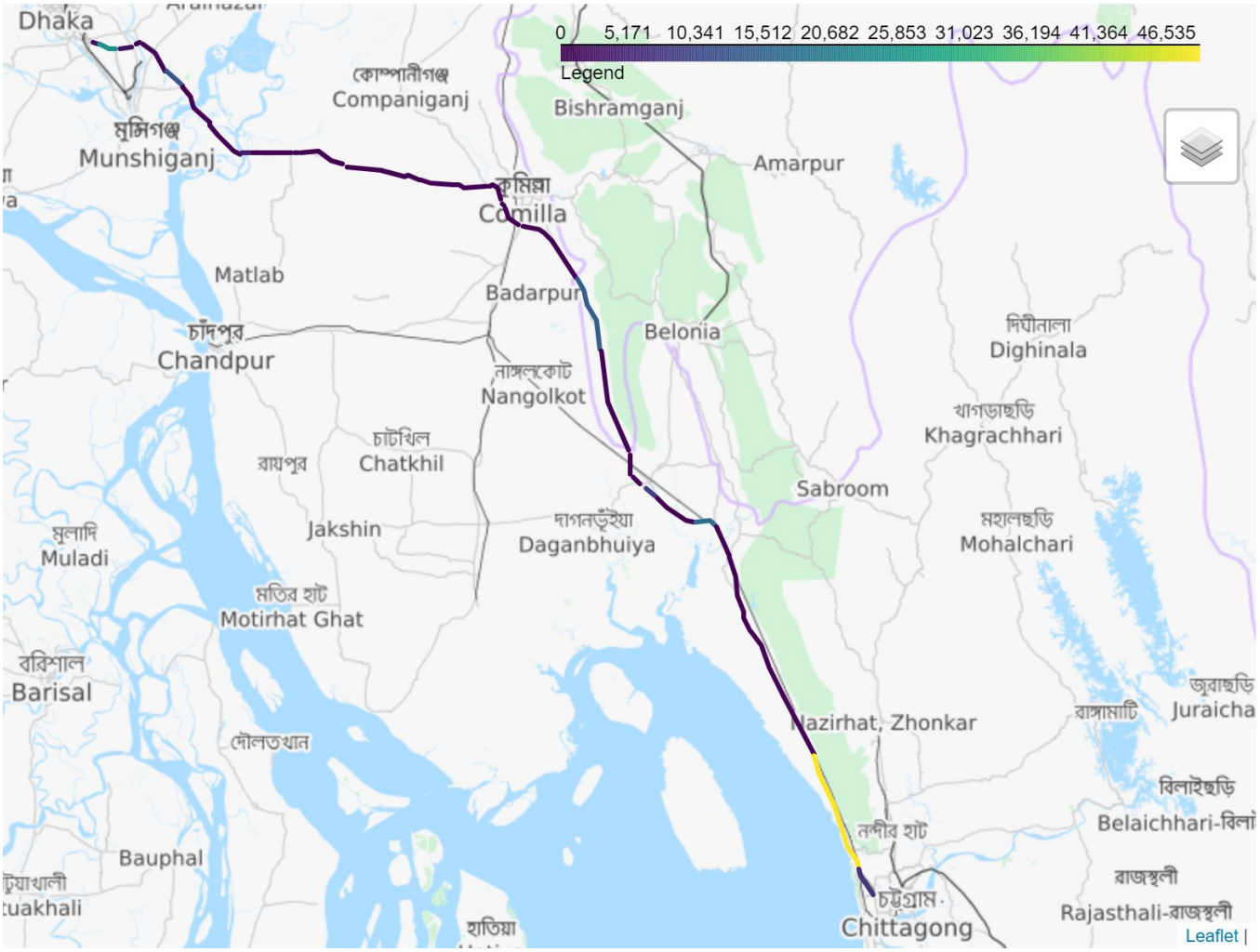
Assignment 4: Distributed Components



Group 16

Contributors:

Brennen Bouwmeester	4446461
Omar Quispel	4107950
Irene Schmidt	4288009
Kevin Su	4438108

Table of Contents

Guideline

Figure 1 illustrates the different processes that take place in this assignment. The names in the oval shapes can also be found in the Python file “Preparing simio input.ipynb”, each being the title for a part of the data preparation. The files on traffic and infrastructure are prepared to be used in Simio, which accordingly turns this input into output. This output is sent to a MySQL server every 60 minutes, which is then visualized using the Python script “Visualization_SQL.ipynb”.

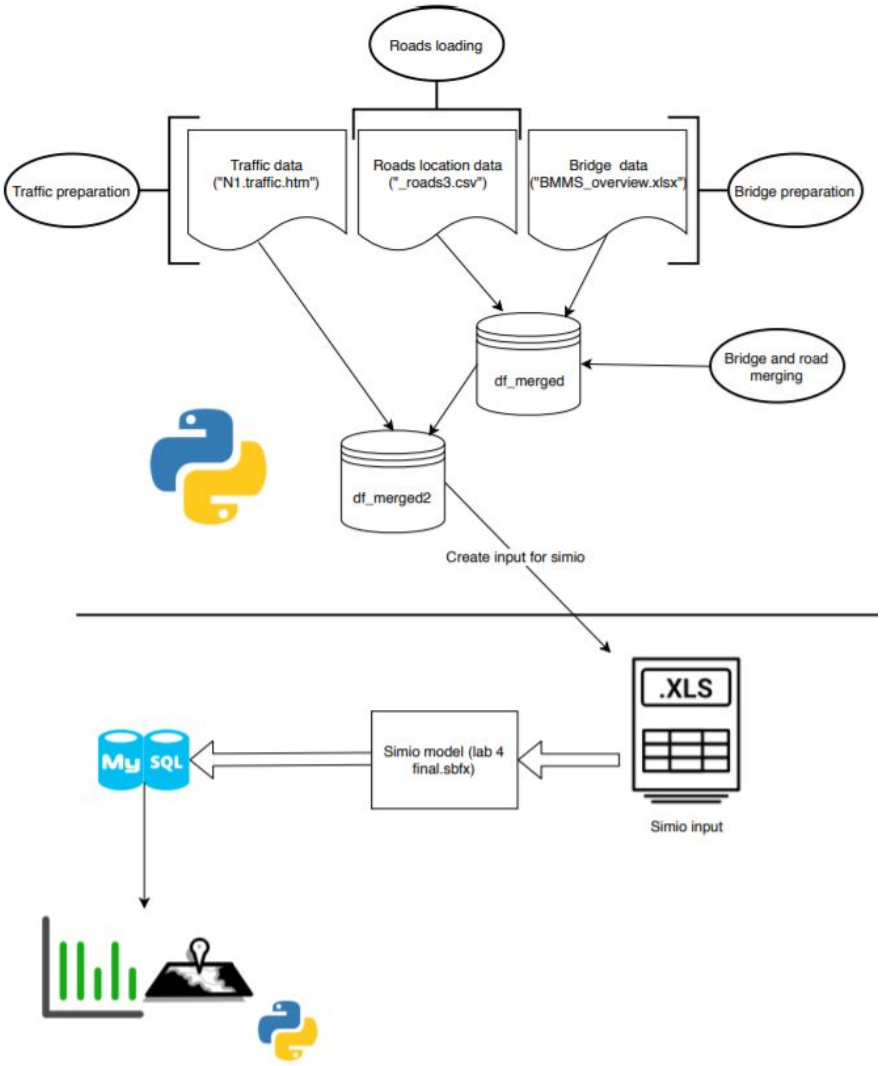


Figure 1. Illustration of the different processes in this assignment

Overview supporting files and instructions:

Python Data and Scripts:

- Visualization_SQL.ipynb
    - FullRunSegments.csv
    - _roads3.csv
    - BMMS_overview.xlsx
    - N1.traffic.htm
- Preparing simio input.ipynb
    - _roads3.csv
    - BMMS_overview.xlsx
    - N1.traffic.htm

Simio model and data:

- Group16 Assignment 4 Simulation Model.spfx
    - CreateVehicles.xlsx
    - SimioComponentGeneration.xlsx
    - segment_import.xlsx

Database Schema:

- epa1351group16.sql

1. Introduction

When analyzing the critical infrastructure in Bangladesh, time and flexibility are important in the decision-making process. In order to help this process, the report will describe the integration of the following components: data cleaning processes in Python for Simio, communication engine as it relates to MySQL, simulation in Simio, and visualization in Python. This will achieve visualizations of key performance indicators on dynamic output data of the model.

The steps outlined in the report follow the flow in Figure 2. First, the cleaning process of data for the model will be discussed. This step achieves the necessary format needed to instantiate a Simio model from an Excel file. Secondly, the model simulation will be discussed. Thirdly, the output data of the model will be stored in a database. MySQL will be used as a database server. Lastly, this database will be connected to visualize the output in Python.

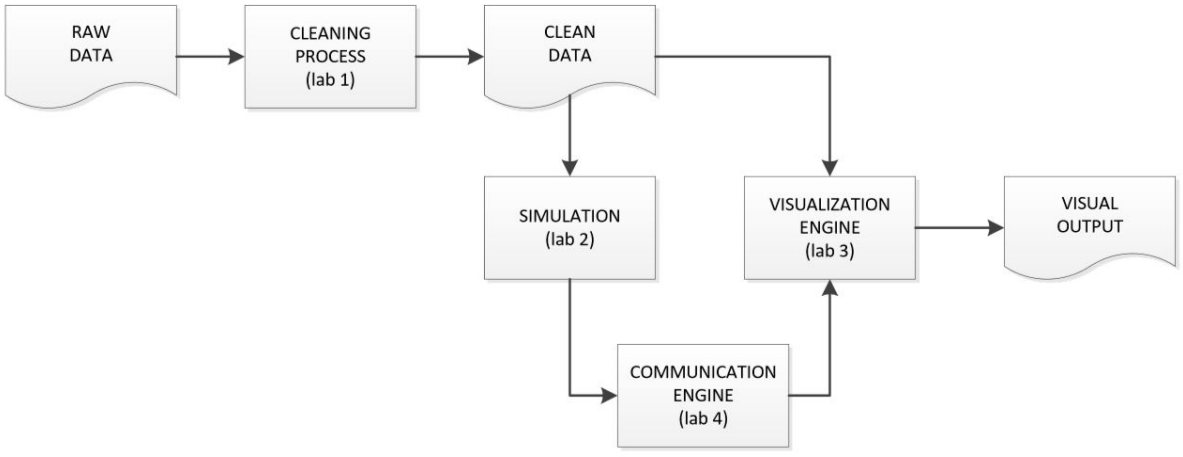


Figure 2. Pipeline for our simulation (Assignment 4: Distributed Components)

2. Data Cleaning Process

Before anything useful can be outputted from a Simio model, and visualized in through a Python script, it is important to clean the input data. In the Python file “Preparing simio input.ipynb” excel file used in the Simio model is generated. By combining the traffic data (N1.traffic.htm) for each of the segments of the N1 road with the lrp data (_roads3.csv), an overview is created for the traffic on the different parts of the N1 road. By merging this combined dataframe with the bridge data (BMMS_overview.xlsx), some parts of the segments can be identified as special points of interest, namely bridges that can break down. After some more cleaning a dataframe is created that holds data for all interesting points on the N1 road, including bridges and their probability of breaking down, but also the amount of traffic that passes each of these points. To smoothen the process within Simio, the amount of traffic is divided by 40 to get less entities in the system. This data can be used by the Simio model to simulate traffic on the N1 and create useful output. To make the Simio model even smoother, most of the non-interesting lrp points are removed from the data. Only bridges, and start or endpoints of the segments are kept in, because these are the only points where the amount of traffic changes or the traffic can encounter delays due to breakdowns. Figure 3 shows an overview of the data frame that is used as input for Simio.

	ObjectType	name	lat	lon	length	condition	AddonProcess	AddonProcess2	NextNode
0	Road	N1_LRPS_Others	23.706028	90.443333	11.3	A	GeneralTravel	DestroyVehicles_Road	N1_LRPSb_CrossRoad
1	Road	N1_LRPSb_CrossRoad	23.702778	90.450472	11.3	A	GeneralTravel	DestroyVehicles_Road	N1_LRP001_KmPost
2	Road	N1_LRP001_KmPost	23.702139	90.451972	11.3	A	GeneralTravel	DestroyVehicles_Road	N1_LRP001a_Bridge2
3	Bridge	N1_LRP001a_Bridge2	23.698739	90.458861	11.3	A	InteractBridge	DestroyVehicles_Bridge	N1_LRP004a_SideRoad_Right
4	Road	N1_LRP004a_SideRoad_Right	23.693805	90.480527	6.6	A	GeneralTravel	DestroyVehicles_Road	N1_LRP004b_Bridge2

segment_x	position_in_segment	CreateTrucks	CreateBusses	CreateCars	CreateCycles	DestroyTrucks	DestroyBusses	DestroyCars	DestroyCycles
segment1	0.0	33.0	23.0	22.0	7.0	0.000000	0.0	0.000000	0.0
segment1	2.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0
segment2	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0
segment2	1.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0
segment2	2.0	0.0	1.0	0.0	3.0	0.393939	0.0	0.227273	0.0

Figure 3. Input data frame for Simio

3. Simulation

3.1 Model explanation

The simulation model from assignment 2 was used as a basis for assignment 4. The main change is that the entity creation/destruction is now based on processes instead of sources/sinks. This is to accommodate the creation and destruction of vehicles across the N1 between Chittagong and Dhaka based on congestion data. Entities are created at the start of the run and every five minutes after that. Note that the different kinds of traffic also have a different speed on the roads. This is done at Dhaka and the start of every segment where more entities are needed than at the previous segment.When a reduction in entities is needed a destroy is used which is based on the ratio of entities that need to be destroyed from the previous segment to the next segment to attain the correct amount. For both creation and destruction of entities new properties were added to the *Road* and *Bridge* objects.

In the data cleaning process it was discussed that the number of entities were reduced, both in general amount as well as to to reduce to the number that had to be created every 5 minutes in Simio. Even with this reduced number, the model still ran very slowly. As such the timer to create vehicles now triggers every hour instead of every five minutes. This changes the number of vehicles in the model significantly, however it helps show that the component linking works as the model doesn't start slowing down a lot until deep into day one, instead of after an hour of run time.

Dummy nodes from the 'segment' object type were added in the floor label just north west of Dhaka. These are used to artificially store information on the segments instead of creating model variables.The floor label also contains a dummy node called 'NewVehicle' which is used as an initial value for congestion data collection in the model.

Since we did the bonus the user is free to decide which bridges to break using the Python file 'PreparingSimioInput.ipynb'. In the current data file 'SimioComponentGeneration.xlsx the following bridges were broken (using our naming structure of road-lrp-type):

- N1_LRP001a_Bridge2
- N1_LRP004b_Bridge2
- N1_LRP014a_Bridge2
- N1_LRP024_KmPost
- N1_LRP231a_Bridge2

3.2 Model Verification

Basic verification was performed to make increase confidence in the workings of the model. The different steps will be discussed briefly. Entity routing was tested by moving nodes off the linear path to make sure that entities were going were they should and not just to the closest node. Tests were also performed to make sure that the delay was performed correctly on the entities as they tried to cross broken bridges. Trace tests were also performed to check the correctness of the various states and output statistics. A problem with the way the model is created, is that the model needs a warm up time in order to act

realistically. Because traffic is only created at certain segments, most of the segments are empty in the beginning of the model.

3.3 Bonus Assignment One

In the last few cells of the Python file “Preparing simio input.ipynb”, the possibility is given to choose which of the bridges break down. Using an “input” function, the user is asked to give the name of the bridge that should breakdown over and over again, until the user decides that enough bridges have been broken down. This information is assigned to each bridge in the Simio model as a property in the bridge object called *Breakdown*. When an entity arrives at a bridge it’ll use the bridge’s *Breakdown* variable to skip the broken bridge process (Breakdown=0) or to find out how long to delay itself (Breakdown=1).

breakdown
0
0
0
0
0
1

Figure 4. Column that indicates if a bridge breaks down

4 Communication Engine

For the communication between Simio and Python the communication engine MySQL Workbench is used. With this workbench the table in Figure 5 has been created. Simio can send data to this table and Python can extract this data. The data that is send as output is the congestion per segment per vehicle type and the total delay time incurred by vehicles per segment due to broken bridges.

To establish a connection between Simio and the MySQL workbench a Db connect element was added. For the data communication two processes were created, see Figure 6. In the process OnRunInitialized a Db Execute step is used to delete any old data in the MySQL table at the beginning of each run. In the other process the DbRead and DbWrite steps are used to send data to MySQL. Figure 6 shows what was added to the DbRead and DbWrite steps. To move the data from MySQL to Python, a script is used which connects to the database using relevant user information. Data is then read in intervals which is discussed in chapter 5.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
RowID	INT(11)	☑	☑	☐	☐	☐	☐	☐	☐	
Name	CHAR(15)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
time	DATETIME	☐	☐	☐	☐	☐	☐	☐	☐	NULL
NrTrucks	MEDIUMINT(9)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
NrCars	MEDIUMINT(9)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
NrBusses	MEDIUMINT(9)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
NrBicycles	MEDIUMINT(9)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
TotalBridgeDelay	DECIMAL(10,2)	☐	☐	☐	☐	☐	☐	☐	☐	NULL

Figure 5. columns created in MySQL Workbench

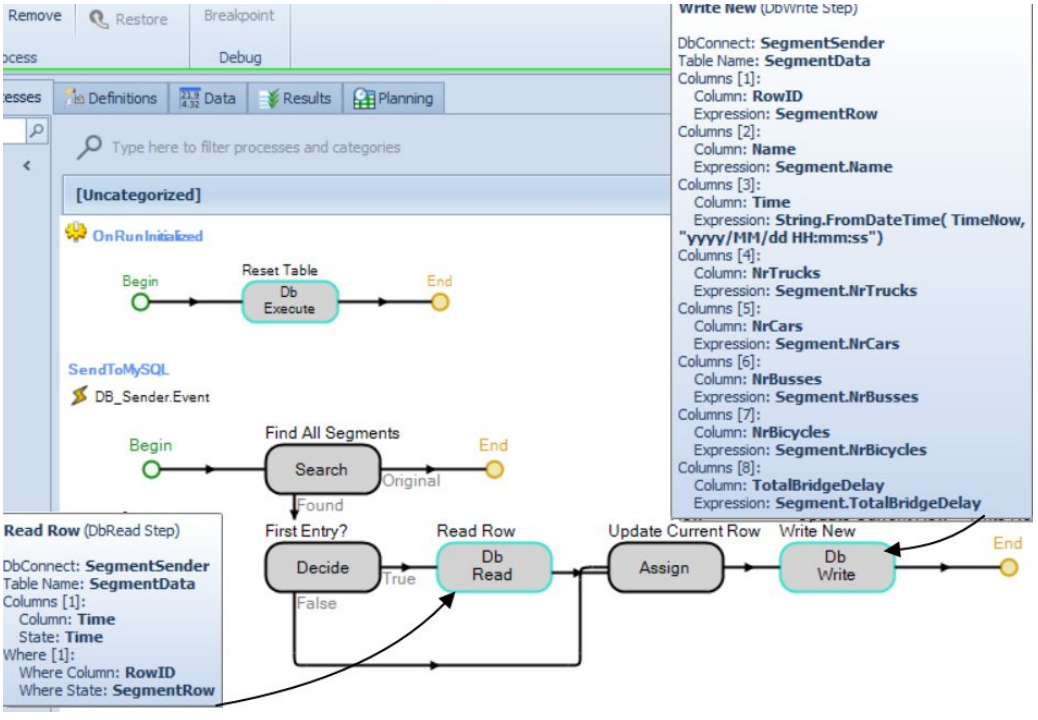


Figure 6. Simio processes used for data communication

5 Visualization Engine

5.1 Key Performance Indicators

To communicate the results of the model it is important that the visualization is clear and precise for the purpose at hand. The data loaded in from the MySQL server, is being further cleaned and shaped in the Python file “Visualization_SQL.ipynb”. Every time step new data is loaded, but only the last row of the data is taken into account for the visualization. The comments in the Python file explain the cleaning from SQL output, to the visualizations.
After this process, the following KPI’s are visualized:

- Delay time per segment
- Traffic density per segment

In Figure 7 these KPI’s are shown. In the Python file these visualizations are updated every 5 seconds.

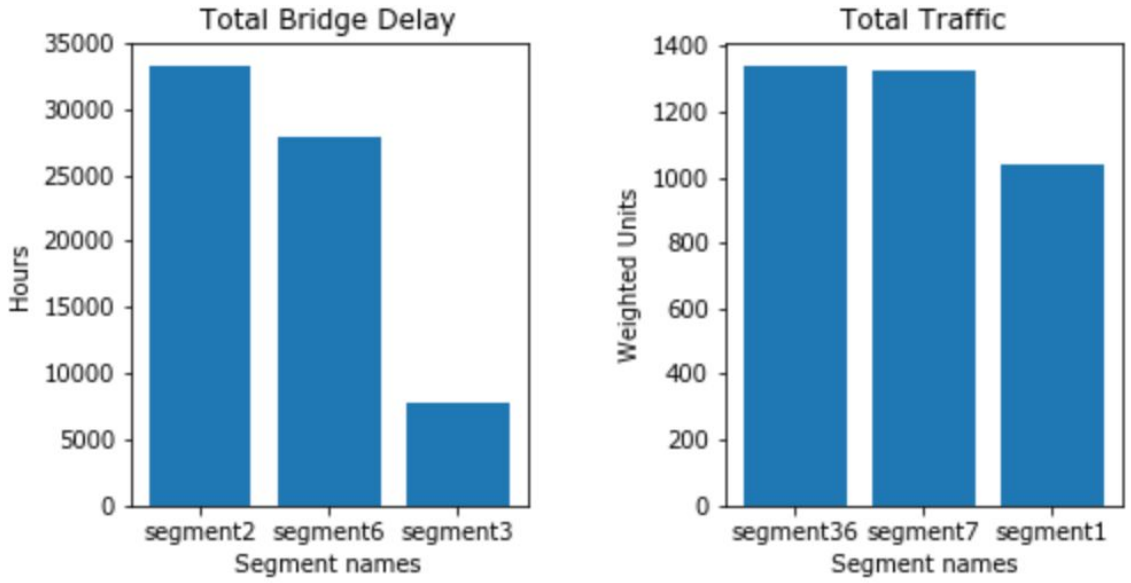


Figure 7. Total Bridge Delay and Total Traffic

5.2 Bonus 2

To be able to replay the development of the segments over time after the simulation is done, an additional chapter (replay process) has been added to the Python file “Visualization_SQL.ipynb” that replays the process in a faster way. This can happen because this function doesn’t have to wait for new data to come in, as this data is already there. The speed at which the visualizing happens, can be changed by the user.

6 Limitations and recommendations

In this chapter the limitations and recommendations that lead from those shortcomings will be explained. By linking the processes done in the previous assignments, the usability of the assignments has increased. The state of the different segments of the N1 can be tracked while the Simio model simulates the traffic through the road.

Although the model simulation and output visualization are interlinked and can run at the same time, more automation could be implemented to make the system even more efficient. For example, when also the data preparation would be linked to the automated process, the whole system could be automated. Then an interface could be built that has room for levers and buttons to destroy bridges by hand, which shows in real time what happens to the system.

If we could get our hands on real current traffic data, the system could also be used to monitor the road system and anticipate probable problems in the network. To improve the automation even further, a real SQL server could be used, which means all the different software packages could be run on different computers, which could be useful when the full road network is taken into account.

To further improve the interface, notifications could be shared only when a (future) critical delay takes place in the system. This way the system could run 24 hours a day, but only notify the users when something interesting happens.

Another possible improvement is the addition of other roads to the system, as well as links between the roads. If this is included, traffic can use alternate routes when it is known that bridges aren't available. In the current model all traffic will focus on a single road blindly, yet it may be more efficient to take an alternative route to their destination. It could also be of value to incorporate the repairing of bridges before the end of the run. If this is relevant in practise, it may reduce the relevance of certain smaller bridges.

Right now even working with just the N1 road the model runs very slowly. To work on the whole road, network optimisations should be made. Improvements are likely found in how certain processes are defined, to make the runs faster. Currently it is difficult to make iterations or even quick tests as the model runs relatively slow.