DELFT UNIVERSITY OF TECHNOLOGY

FACULTY OF TECHNOLOGY, POLICY AND MANAGEMENT

MASTER THESIS PROJECT REPORT

Assessing The Impact of Capacity of Depots and Vehicle Schedule in Transportation Systems

---

Author:

Jing CAI

Supervisors:

Committee Chairman:
Prof. A. VERBRAECK

First Supervisor:
Dr. M.D. SECK

Second Supervisor:
Dr. M.P.M. RUIJGH-VAN DER PLOEG

External Supervisor:
H.J. VELDHOEN (HTM)

August, 2011

Acknowledgments

This report presents the result of my internship at HTM Personenvervoer NV from March 2011 to August 2011, which is a part of the final project to obtain the degree of MSc (Engineering and Policy Analysis) at the faculty of Technology, Policy and Management at Delft University of Technology.

It is never easy if people want to achieve something. For me, there have been failing, struggling, learning and, fortunately, improving in this long process. Sometimes, I have to work so hard that I have even forgotten what I am working for. Looking back now, I realize that the greatest days are not the ones when I worry the least, but the ones when I try the hardest, and when I know that is it not what I am rewarded but what I have learned and experienced matter the most.

I have received great help from many people in this process. First, I want to express my sincere gratefulness towards my first supervisor Dr. Mamadou Seck. Thank you for your help, patience and encouragement to me. Without them, I could never have achieved what I have today. It does not matter what is the final result, I am proud of my work, I feel lucky to be your student, and I am grateful for what you have taught me and what you have done for me. To my chairman of committee Prof. Verbraeck, you have enlightened me with your widsom and guided me with your insightfulness. To my second supervisor Dr. Ruijgh-van der Ploeg, thank you for your providing your insight into the problem and granting me a broadened perspective into the research. To my external supervisor Hilbert, thank you for supporting me in every possbile way and I hope what I have done can be of some help for your work. To Yilin, thank you for sitting with me for hours and hours for the seemingly endless debugging and wish you all the luck on your future research.

I want to decicate this thesis to all of my friends from EMIN and EPA 2009 - 2011. You all are my family, and you gave me everything when I am away from home. To Shi Nan, for you craziness and precious friendship. To Tewes, for you recklessness. To Enrique, for putting me in your acknowledgement and to be in my defence. To Shalini, for you thoughtfulness. To Pablo, Marija, Yasir and Angela, for the great time we had together in Delft and Madrid. To Julia, for talking to me in Engligh. To Tombek, for your comprimises. And to my dearest parents, for your endless love and conditionless support. There are just too many people to thank, yet I have to write the last sentence of the two years in the end. I will never forget anyone of you, and I will follow you everywhere until the end of the world.

Cai Jing

Delft, August 2011

Executive Summary

A well designed public transportation system can provide convenience to the public, increase the profit of the transportation company, and reduce the cost to the society as a whole. The depot is an important part in the study of transportation systems. The choice of the capacity of the depots is a strategic decision. It will constrain the possible tactical decisions (such as vehicle planning) for a long time, due to the life-span of the infrastructure. It is therefore important to study the impact of the options before they are implemented.

The transportation company now is facing the problem that there is no appropriate tool in the decision support system to assess the capacity of the depots and the vehicle planning on the deadhead-kilometer (non-value added trips) and the robustess of the service.

The simulation model is an appropriate method for the main part of the study. It can be used to evaluate the key performance indicators of deadhead-kilometer and average delay time, to study the complex interactions among the different elements, and to improve the understanding of the behaviors of the transportation system. The simulation model is useful for evaluating both strategic decisions and tactical decisions.

The analyzing tool is designed following the methodology of systems engineering. First the requirements were gathered and translated into specification. The design was then conducted to fulfill the requirements on the conceptual level. Specification of the key classes were then concluded from their functionalities and relationships. The detail design came out with detailed attributions and methods of the classes.

The analysis is based on a part of the network which consists of five service lines. According to the analysis, compared to the current design (Scheveningen, Zichtenburg and Lijsterbes), the future alternative with "Remise Zuid" depot and closing down "Lijsterbes" depot achieves 18% reduction on average delay time, but with a cost of 15% of increase in deadhead-kilometer. With the ratings obtained from the policy advisors of HTM, the alternative with "Remise Zuid" outperforms the design of current situation on the overall performance. Though the methodology applies to the entire network, the analysis results so far portaits the impact on these five lines. A more comprehensive study can be conducted with the same method and the input data covering all the operating lines.

Some improvement can be made on the management of depots by a better the training of the employees and a clearer definition of job responsibilities.

Keywords: rail transportation, depot planning, vehicle planning, discrete-event simulation

Contents

Acknowledgments **1**

Executive Summary **2**

Contents **3**

1 Introduction **6**
1.1 The Depot and its Importance 6
1.2 The Research Problem 7
1.3 Research Questions 8
1.4 Research Methodology 9
1.5 Outline of the Report 11

2 Analysis of the Problem Complexity **12**
2.1 Introduction to Transportation Company HTM 12
2.1.1 Functions of the transportation system 13
2.1.2 The *Plan* function 15
2.2 Introduction to the Depots 16
2.2.1 Functions of the depots 16
2.2.2 Some potential room for improvement on management 18
2.3 Key Performance Indicators 18

3 Methods for Studying Transportation Systems **20**
3.1 Decisions on Different Levels 20
3.2 Research Methods for Transportation Network 21
3.2.1 Analytical and optimization models 21
3.2.2 Simulation models 22
3.3 Decisions for the Future 24
3.3.1 Dealing with uncertainty 24
3.3.2 Multi-attribute utility theory 25

4 Discrete-Event Modeling: DEVS Formalism and LIBROS-II Library **27**
4.1 DEVS Formalism 27
4.1.1 Atomic models 27
4.1.2 Ports 28
4.1.3 Coupled model 28
4.2 LIBROS-II Library 28
4.2.1 `AtomicModel` and `CoupledModel` 28
4.2.2 `RailVehicle` 29
4.2.3 `Sink` and `Source` 29
4.2.4 `TrackSegment` 30
4.2.5 `Interlocking` 30
4.2.6 `NodeMessage` 30
4.2.7 `CadModelBuilder` 30

5 Design of the Model **32**
5.1 Requirements of the Model . . . 32
5.2 Conceptual Design . . . 32
5.3 Specification of Key Classes . . . 34
5.3.1 `Depot` . . . 34
5.3.2 `ControlCenter` . . . 37
5.3.3 `RailVehicle` . . . 38
5.3.4 `TrackSegment` . . . 41
5.4 Detailed Design . . . 42
5.4.1 Assumptions and simplifications . . . 42
5.4.2 Preparation of input files . . . 42
Service Timetable . . . 43
Vehicle schedule . . . 43
Optimization solver . . . 43
Vehicle routing . . . 43
Initial value . . . 43
5.4.3 Preparation of input AutoCAD file . . . 44
5.4.4 Model construction . . . 44
5.4.5 Initialization . . . 46
5.4.6 Normal running . . . 48
5.4.7 Disturbance . . . 49
5.4.8 Data output . . . 50
5.5 JAVA Code Implementation . . . 51

6 Verification and Validation **56**
6.1 Verification . . . 56
6.2 Walkthrough . . . 57
6.3 Compare the model results with the real data . . . 58
6.3.1 Validation of distance . . . 58
6.3.2 Validation of time . . . 59

7 Evaluation of the Impact of Capacity of Depots and Vehicle Planning on Key Performance Indicators **61**
7.1 Computation of Key Performance Indicators . . . 61
7.2 Comparison among Future Alternatives of the Capacity of the Depots 61
7.2.1 Input data . . . 61
7.2.2 Results of the impact on the key performance indicators . . . 63
7.2.3 Combined results with weights of the criteria . . . 65
7.3 Comparison among the Alternatives under Uncertainty . . . 66
7.3.1 Uncertain factors . . . 66
7.3.2 Analysis under uncertainties in weights of criteria . . . 67
7.4 Vehicle Allocation . . . 68

8 Findings and Conclusions **71**
8.1 Answers to Research Questions . . . 71
8.2 Conclusions and Recommendations . . . 73
8.3 Reflections . . . 74
8.4 Future Study . . . 75

References **77**

Appendix **80**

A Detail Description of Depot Zichtenburg **80**
A.1 Detail of the Plan Function 82
A.2 Accidents 83

B Documentation of LIBROS-II and New Development **84**
B.1 Atomic Model 84
B.2 InfraComponenet 84
B.3 trackEntity 84
B.4 Depot 85
B.5 RailVehicle 89
B.6 VehicleGenerator 94
B.7 DataOutputer 94
B.8 LibrosModel 94
B.9 CadModelBuilder 94
B.10 Constructing speed limit 96
B.11 Special attention with programming with LIBROS-II 98
B.12 Input files 98
B.12.1 Vehicle Schedule 98
B.12.2 Optimization model solver 98
B.12.3 deadheadRoute.xls 98
B.12.4 initialValue.xls 98

C Design **102**

D Verification and Validation **104**
D.1 Validation: Trip Distance 104
D.2 Validation: Statistical Test 105

E More Results of the Analysis of the Impacts **107**
E.1 Detailed Results of the Analysis 107
E.2 Statistical Test of the Results from the Model 110

F Examples of Deadhead Routes **113**

1 Introduction

Public transportation systems play an important role in many aspects of a society. For the public, it can provide more options of transportation with great convenience. People can go to different places without owning a private car and with a lower cost. Nowadays, public transportation system has been an important part in people's daily life. People rely on public transportation to a great extent for many activities. To the society, a well designed public transportation system can reduce the number of private cars used, reduce fuel consumption, atmospheric pollution and traffic congestion, and the cost of the society as a whole.

Public transportation systems are subsidized by the government in many places in the world. So it is in the busy city of The Hage. HTM is a Dutch company that offers collective passengers transport services for the city of The Hague and the surrounding area (Haaglanden). The transit authority Stadsgewest Haaglanden finances most of the public transport services delivered by HTM. The transit authority defines the key performance indicators, and participates in cooperation in the decision-making process at the strategic level of HTM.

An improvement on the design of public transportation can increase the profit for HTM, reduce the economical cost of the transit authority, and reduce the social cost of the city as a whole. Therefore, it is vital to study the public transportation system in order to identify real opportunities to improve performance and to further achieve the benefits. In this chapter, the research background of the project, research problem and methodology are briefly introduced.

1.1 The Depot and its Importance

Depots are important parts in the analysis of transportation systems. Generally, depots are where vehicles are maintained, most of them are kept over night and from which they are dispatched for service. Operations such as cleaning and maintenance by special staff are essential to keep the vehicles in good condition for service for a long life-span. It is for this reason that all the vehicles need to be scheduled to go to depots regularly for those operations after some time of running.

The capacities of the depots limit the number of vehicles that can be in the depot at the same time. Vehicle schedule (or vehicle timetable) specifies the time of service trip, returning to depot and other operations inside the depot for each vehicle. It is affected by the capacities of the depots because when one depot has reached its capacity for the number of vehicles kept over night, other vehicles can only park in other depots.

Usually the depots are not the starting points where vehicles begin their exploitation trips, because it is too costly to have a depot at each starting point of the service. So the vehicles need to travel from depots to the starting point in the morning, the other way round late at night and sometimes during the off-peak hours in the day. This distance that the vehicles travel without providing service is called deadhead-kilometer, such as the trip from the depot to the beginning of the line. This process does not provide any profit, but still costs other resources including electricity, driver, and maintenance staff and material. For HTM, A rough estimation of the total cost of deadhead-kilometer is 3.7 million Euros per year for the currently situation. In the face of economy recession, any chance of reducing cost is important to the companies. Meanwhile, the time it takes to travel this distance also slows down the speed of the reaction of the network to disturbances. This is because for every minute that a standby vehicle would use to travel to the location of disturbance to resume the service, there are many travelers

waiting at the stops for the vehicle. As a result, the deadhead-kilometer is useless, undesirable and should be minimized.

Another reason that the depots are important is that they affect the robustness of service of the system. A robust system will not vary much when disturbances occur and can resume to its normal state within a short period of time. For example, when a disturbance happens, the delays of service can greatly affect satisfaction of the travelers with the service. Besides, the transit authority dictates the percentage of service that should be fulfilled. If this percentage of not achieved, extra penalty costs will be levied onto the transportation company. These factors are affected by the locations of the depots, because the farther the depots are from the location of the disturbance, the longer time it will take for the substitute vehicles to come. The capacity of the depots also limit the number of stand-by vehicles in the depots, thus affect from which depot can a stand-by vehicle be dispatched.

Due to the development of the tram network and the development plan for the city of The Hague by the Municipality, HTM decided to build a new depot. The new depot will have a certain impact on the system. The company is also facing the choices of closing down some old depots and centralizing maintenance areas in different depots. These options give the company some space to rethink the impact of the depots on its overall management activities and to improve its service level.

For HTM, it is important to measure and monitor those criteria when considering a decision for the future. As a result, the study of design and operations related with depots is of paramount importance to improve the quality of service and the profit of the company.

1.2 The Research Problem

As has been discussed before that depots are important, we should study the impact that the depots have on the transportation system before taking decision about the development plan of the depots. HTM has already developed a simulation model with TU Delft, a description of which will be given in section 4.2. However, the simulation model currently used by HTM cannot meet the demand to provide the information for such study due to several reasons.

First, the current simulation model is unable to represent the operations of vehicles in reality. This is because it only considers the exploitation process: vehicles are generated when starting their exploitation process right before the beginning of the line and are eliminated after finishing their job in the model right after the end of the line. The absence of depots can make the results of the simulation model greatly deviate from what happens in reality, such as delays of service. For example, if a vehicle is delayed in the previous exploitation process or maintenance in reality, it will likely be delayed in the next exploitation process. If a delay or any other change happens to the operations inside the depots, it can also cause delay to the exploitation processes outside the depots. But in the current simulation model, this temporal dependency between consecutive service and/or maintenance processes is not presented properly. Hence, it is difficult for the current model to represent the operations of vehicles and to assess vehicle planning.

Second, deadhead-kilometer is an important performance indicator for the transportation system and should be assessed when studying depots in transportation system. However, in the current model, all vehicles do not travel the deadhead trips as they do in reality, thus it is impossible to measure the indicator of deadhead-kilometer in the model.

Third, the model is unable to reflect vehicle schedule. Currently the service is composed by each "trip", where the vehicle is generated at the beginning of the trip and deleted when the trip is finished. As a result, we also cannot tell which vehicle belongs to which depot and how the vehicles from a certain depot perform in the network. Vehicle planning is an important aspect to reflect the design, management and operation of depots, as it is constrained by the depots' capacity and the operations of the vehicles inside the depots are designed according to vehicle schedule. The absence of this element makes it difficult to measure how each depot affects the system.

Fourth, the current simulation model only represents the current situation of the system, instead of the situations when the new depot will be in use. An operational model, such as the current simulation model used by HTM, can be used to provide information of some performance criteria to support strategic decisions (such as choosing the capacity of a new depot) or tactical decisions (such as timetable design and vehicle planning). It can also be used to test the designed vehicle schedule. However, in order to support such a decision, whose influence will last for decades, it is important that the analysis tool can adapt its assumptions to include uncertainty in the study. This is because we are not sure of the value of many assumptions in the future, and the change of those assumptions can affect the optimality of the decision. For example, operation times inside depots are not the same every time and could also change significantly due to change of operations. If there will be some change of the network in the future, the calculation of deadhead-kilometer and delays of service will also vary, thus different location and capacity of the new depot might be favorable. Only with such information, it is possible to design a robust solution that can guarantee the performance of the system under most circumstances.

As a result, the main research problem is that **the company lacks a tool in the decision support system to assess the capacity of depots and the vehicle planning related to the depots on deadhead-kilometer and on robustness of service**.

1.3 Research Questions

In order to solve the problem, we can design the tool through answering the questions:

1. What are the current situations of the transportation system, which kinds of impacts should be measured and what data is available for analysis?
2. What methods can be used to assess the performance of transportation systems and which one is the most appropriate in this study?
3. How to design the tool to assess the impacts?
4. How to validate the tool to ensure it reflects the reality?
5. What are the impacts of vehicle planning among the depots on deadhead-kilometer and average delay of service?
6. What are the impacts of capacity of depots on deadhead-kilometer and average delay of service?
7. Which types of uncertainties should be considered for a decision for the future? Which ones can be dealt with in this study, and how do they affect the analysis result?

1.4 Research Methodology

The system discussed in this section is the system of the analyzing tool, which will be the product of the research project and used for the analysis of real problems. Analyzing tool is a complex system, which consists of various sub-systems, such as analyzing, data processing and evaluation of sub-systems. The design problem of this tool requires approaches of various disciplines and trade-offs to meet different requirements. "The objective of systems engineering is to see to it that the system is designed, built and operated so that it accomplishes its purpose in the most *cost-effective* way possible, considering performance, cost, schedule, and risk." [1] "It ... recognizes that any system has a number of objectives ... The methods seek to optimize the overall system functions according to the weighted objectives and to achieve maximum compatibility of its parts." [2] Having said above, it is clear that systems engineering is suitable to address the problem defined in this project.

The life cycle of systems engineering is divided into three phases (Definition, Development and Deployment). One type of life cycle model with different steps within each phase defined by Sage is shown in figure 1.1. According to Sage, it "assist clients through the formulation, analysis, and interpretation of the impacts of proposed policies, controls, or complete systems upon the perceived needs, values, institutional transactions of stakeholders... " [3]

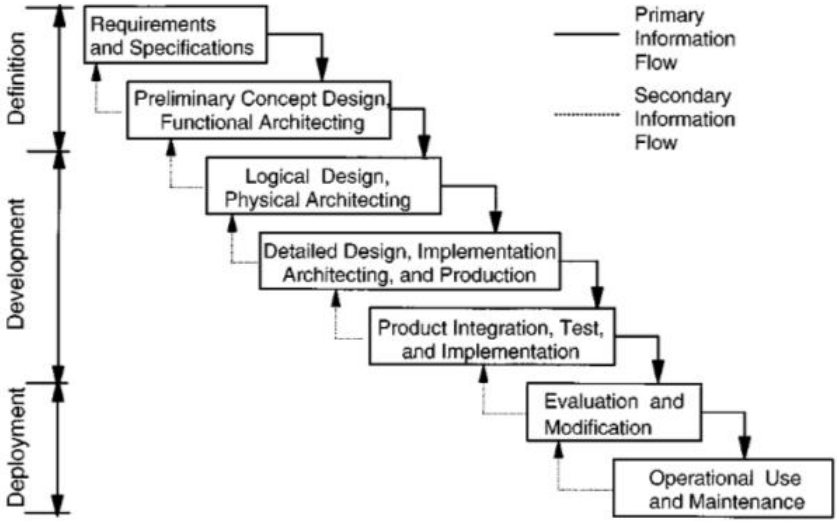


Figure 1.1: Systems Engineering Life-cycle Processes [3]

1. Requirements and specifications of the system

   From figure 1.1 we can see, for a design problem, systems engineering methodology starts with looking at the requirements collected from interview with the problem owner, which is the most important part of a design project and should be always kept in mind of the designer. Interview with the people involved in the problem, field study are used to answer research question 1.

2. Overall structure of the analyzing tool

Preliminary concept design is firstly conducted from a holistic perspective in order to look at the problem of the entire system which combines interdisciplinary approaches. A deeper understanding of the system is required here, so field study and interview with experts will be carried on. By interview with the problem-owner and experts, the scope and boundary of the model is define. Requirements and specifications from the previous step are translated into input, output and constraints of the model. Factors and assumptions involving high uncertainty and high impact on the system should be found out through interview with problem-owner and experts and literature research, and strategy of including uncertainties in the study is defined in this step.

3. Logical and detail design

   In the system development phase, logical design and physical architecting is performed to design the system focusing on the overall structure and architectural style of the model and interactions among different sub-systems. In detail design, it examines each design details and makes sure of the proper function of each sub-systems. First, study of the current simulation model, including the important configuration and interface, is done to make sure that the new model complies with it. Second, field study and data collection are conducted to define and calculate parameters needed in the model, such as time of operations inside depots. Strategy to deal with uncertainties is also taken care of in this step by providing options to change parameters involving uncertainty. Third, the depot module is designed, and the internal logic also has to comply with timetable for travelers and with constraints of capacity. Fourth, the scheduling rules of vehicles to depots and in case of accidents need to be specified and translated into the model.

4. Product integration, test and implementation

   In the last step of development phase, the sub-systems are integrated and tested if they are working well together and with the current simulation model of HTM. Necessary changes are made if there is any conflict or inconsistency. The first step is to design the interface between the depot module and the current simulation model. Second step is to obtain output of data related with deadhead-kilometer and delay in case of accidents, which should be defined in the entire model, is designed. Third step is to calculate the statistical data from the output, which are useful to support decision-making.

5. Evaluation and deployment

   In the system deployment phase, evaluation and modification is conducted to test if the analyzing tool can accomplish the defined objectives well enough or not. Sub-question 4 is answered in this step to perform the test. First literature research is conducted to look for useful and relevant methods for evaluating the model. Afterwards, real data is collected and input into the model to generate analysis results. Then evaluation methods are used to test if the model is working well enough or not. If the model does not perform well, necessary change needs to be made, and then it will be evaluated again until its performance reaches standards.

6. Operational use and conclusion

A case study is conducted to give an explanation of the model and its analysis results. Data analysis is firstly used to obtain information that is useful for later analysis. In order to deal with uncertainty, and analysis is performed to assess the output of criteria when uncertain factors concluded. Afterwards, all the results are integrated together to give an representative and useful result to reflect the level of uncertainty that shows in the transportation system. In the end suggestions and conclusions are given.

The results from a later activity may require a revision of a previous activity, such as change of assumption or adding more detail in the requirements and specification. Then the *secondary information flow* is formed, as indicated in figure 1.1, the previous activity is revised, and the change is reflected in the change of the later activities.

1.5 Outline of the Report

This paper is organized as follows. Chapter 1 gives a brief introduction of the problem background and the research methodology. Chapter 2 introduces the readers to the current situations in the company and gives a closer look at the problem at research. In chapter 3, the theoretical basis of the study is presented. Chapter 4 introduces LIBROS-II model, which is used to design the tool. In chapter 5, the conceptual design of the model is given. Chapter 6 analyzes the results from the model under the current situations and compares it with the real data collected from the company. Chapter 7 gives an analysis of the model for the decision making for the future. In the end, findings and conclusions of the study are discussed in chapter 8.

2 Analysis of the Problem Complexity

A transportation system is a complex system involving many different elements. In order to study the transportation system, it is important to know its functions, work processes first. This chapter gives a brief introduction to functions and work processes within the company HTM and the depots in its network. The key performance indicators are then selected to indicate the performance of the system.

2.1 Introduction to Transportation Company HTM

HTM operates trams, lightrail and buses. It provides a good connection of transportation in the area of Haaglanden, including The Hague and Delft. In year 2011, HTM owns 187 busses, 147 trams (GTL) and 54 RandstadRail (RR) vehicles. The total traveler-kilometers in 2009 were 288,9 million for the rail vehicles (GTL and RR). Figure 2.1 shows the picture of the network of the trams.

Figure 2.1: Locations of depots in the network

The locations of the depots are also shown in figure 2.1 with red dots. Currently there are five depots in the network of HTM. Three of them have maintenance function: Lijsterbes, Scheveningen and Zichtenburg. When they are not equipped with maintenance functions, they can also be called emplacement. Such places include Leidschendam, Oosterheem. Zwarte Pad is a possible location that could become an emplacement in the future, which is very close the current depot Scheveningen. Remise Zuid is a possible location for a new depot. The locations of these possible locations are shown in figure 2.2.

Currently there are two main types of tram vehicles in HTM. RR (figure 2.3) vehicles are wider and shorter than GTL vehicles (figure 2.4). RR vehicles are more advanced than GTL vehicles in terms of the software system and provides more com-

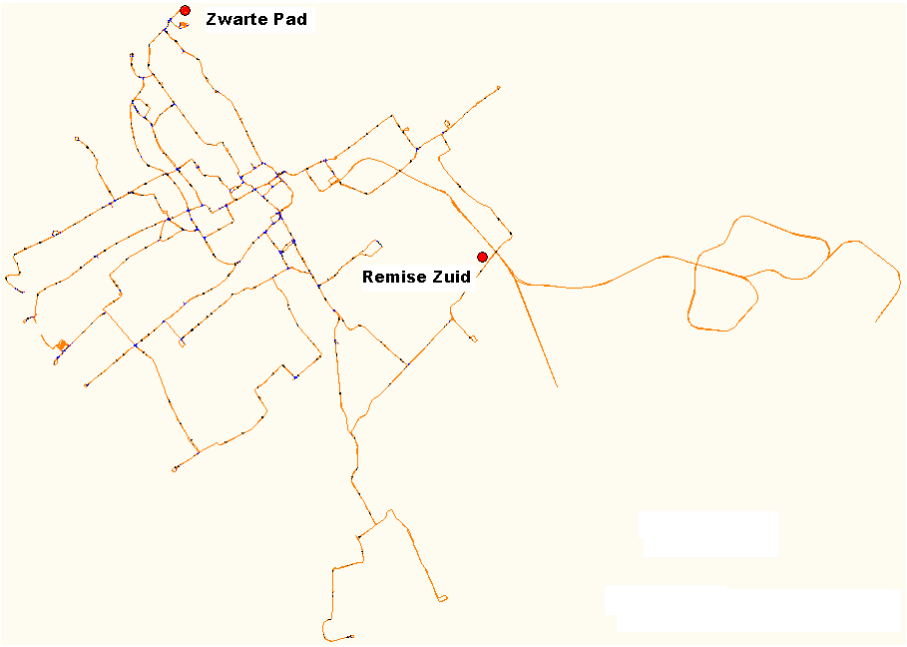


Figure 2.2: Locations of possbiel future depots

fortable space for the travelers. Because of the difference of sizes between types of vehicles, RR vehicles cannot run on most of the old tracks, which were designed for the GTL vehicles. RR vehicles can drive on both directions, while GTL can only drive on only one direction. Due to this reason, the depot designed especially for RR vehicles are different from those for GTL vehicles. In order to improve the comfortability of the vehicles, HTM has purchased more RR vehicles and will stop using some obsolete GTL vehicles in the future. Table 2.1 gives an overview of the number of two types of vehicles that HTM will possess in the coming years.

Year	RR Vehicles	GTL Vehicles
2011	54	147
2012	72	147
2015	112	97

Table 2.1: Number of tram vehicles of HTM

2.1.1 Functions of the transportation system

Figure 2.5 shows the functions of a transportation system. The capacities of the depots are taken as the input of the system. The *Plan* function takes those decisions of the capacities and determines the vehicle schedule, which contains the number of vehicles for each line, the number of standby vehicles, and to which depot and when each vehicle should be scheduled for maintenance.

The location of the depots will affect the distance between the depots and ends of the lines. When scheduling vehicles going to which depot, the company will consider

Figure 2.3: RandstadRail vehicle

whether the depot has maintenance resources (i.e. maintenance function), whether it is closer to where the vehicles start or finish the service trip, and whether the depot has reached its full capacity.

The outcome of the planning is taken as an constraint for the maintenance activities. The plan function also takes into account the maintenance status of the vehicles, which functions as a feedback-loop in the long-term of the *Plan* function. The *Provide Service* takes the vehicle schedule as the input. The *real-time service info* it produces is used for function *Monitor and Control*. *Monitor and Control* will adjust the system according to both the original vehicle schedule and how the system performs in reality, and try to minimize the difference between them and thus the inconvenience it brings to the travelers. In the end, the contingency plan completes a feedback loop from *Monitor and Control* to the function *Provide Service*.

In detail, the function *Monitor and control* (figure 2.6) monitors the real-time service information of all the vehicles. If any disturbance occurs, *Monitor and Control* will receive the type and location of the disturbance and the information of the disturbance vehicle, formulate contingency plan according to specific rules, find standby vehicle, dispatch the vehicle to resume the service as soon as possible according to the published timetable.

The *Provide Service* function uses the vehicle schedule from the *Plan* and provides service to the travelers. The vehicles perform all kinds of operations according to the vehicle schedule from the *Plan* function. If there is any disturbance happen and the vehicle cannot go on with its normal service, it will report to *Monitor and Control*. Then the vehicle waits for the contingency plan and executes it until the service is resumed. The output from this function, *deadhead-kilometer* and *average delay of service* are used as criteria that to evaluate the system.

From the analysis of the functions of the transportation system, we can see that the capacity of the depots, as the input of the entire system, and the vehicle schedule, as the

Figure 2.4: GTL tram vehicle

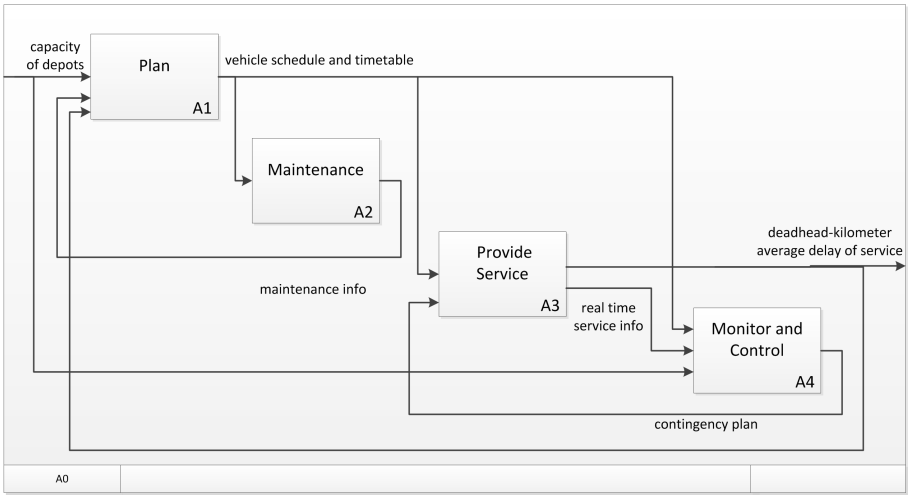


Figure 2.5: Functions of the system

output of the *Plan* function and guidance of many other functions, are very important to the performance of the system.

2.1.2 The *Plan* function

On the strategic level, a decision has been made that a new depot is going to be built. However, the capacity of the new depot, which of the current depots should be closed

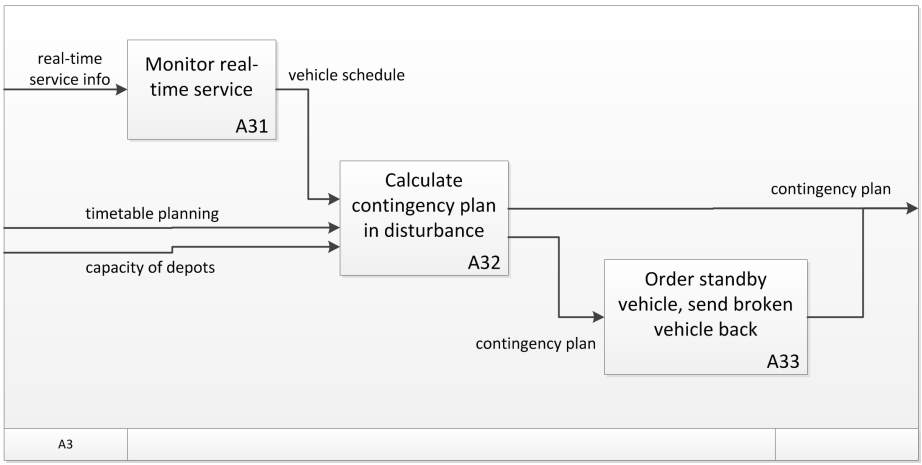


Figure 2.6: Monitor and Control function

or kept are still not clear.

On the tactical level, the current planning process in HTM can be divided into three steps: timetable planning, vehicle planning, and personnel planning. The timetable planning is constrained by the design of the depots, vehicle planning is made according to the timetable. It includes the scheduling of vehicles, that is, which exploitation process is the vehicle scheduled for, and when and to which depot should the vehicle go for maintenance and parking. And vehicle planning influence the personnel planning. At the end, these three steps of planning are interrelated with each other, and all affect the efficiency of the system. The result of these planning process are translated into the daily operations of the system, and thus affect the operational decisions and the performance of the system. The detail of the plan function is introduced in Appendix A.1.

As a result, the daily operational performance of the system is an feed back to the decisions made before. It is an indicator of whether all the decisions on the higher level have been made correctly, and should be used to evaluate and reflect on the strategic and tactical decisions.

2.2 Introduction to the Depots

2.2.1 Functions of the depots

Currently there are five depots in the network of HTM. Three of them are maintenance depots, where vehicle can be maintained and repaired because they are equipped with the necessary mechanics and equipment. HTM is planning to build another depot in the south part of the network, which is called “Remise Zuid” so far. The locations of these depots are shown in figure 2.1 with red dots. The rest are called emplacement, where vehicles can be parked over night and only small and simple operations could be done to the vehicles.

Depot Zichtenburg is the biggest depot in HTM’s network. It is responsible for vehicle planning, contingency plans formulation and major vehicle maintenance. In Zichtenburg, there is a software monitor and control system, which is used to have an overview of the location of all the vehicles inside the depot and to change switches

when the vehicles need to move. The division of the Depot Zichtenburg can be seen from the interface of its control system, which is shown in figure 2.7.

Figure 2.7: Depot Zichtenburg in the controling system

All the vehicle planning is done in depot Zichtenburg the night before. First the vehicle schedules are made about the trips that each vehicle has to fulfill. Afterwards, this schedule is handed over to the maintenance department, who will schedule their operations accordingly, including cleaning, sanding and preventive maintenance accordingly. Then task lists are distributed to each driver of the vehicle. Next day, the drivers follow that task lists, which tell them when to arrive at which stop for which line, and when to go back to the depot. A very important rule for the operations inside the depots is that they should not interfere with the service exploitation. Since mostly operations inside the depots can be finished within a short period of time, and many vehicles can stay in the depot for a long time during off-peak hours (from 10 am to 3 pm and after 9 pm), most of the time this rule can be followed by the operations inside the depots.

The depot is divided into three parts. The first one is to place the GTL trams. Only the first track in this part is equipped with a washing and a sanding machine, which is used for all the washing and part of the sanding operations of the vehicles in this depot. The second part of the depot is used to place the new RandstadRail vehicles. The last part is used for major maintenance. In order to carry out a major maintenance task that needs special tools, the maintenance mechanics have to use a special car to drag the vehicle into the maintenance area. This is because, due to safety reason, there is no power supply for the vehicles inside the maintenance area. For small maintenance, the vehicles can be left in the parking area and the maintenance mechanics can perform their job there.

2.2.2 Some potential room for improvement on management

There are still some problems related with maintenance inside the depots interfering with the service process. During one interview at the control center of Zichtenburg, two similar accidents happened, where the track system inside the depot could not "recognize" the movements of the vehicles, thus treated them as dangerous and closed the interlocking system. As a result, no vehicle could get out or get in the depot for around 10 minutes, which resulted in delay of service from the beginning. The accidents are introduced in more detail in Appendix A.2.

This problem can in fact be solved by a better training of the employees, let them follow the rules of operations, for example notifying the central office before some operations. Besides, the software system can also be improved by, for example, delineating the track system into several parts that can be shut down separated, which will cause less delay if the vehicles need to get out when a problem occurs.

This interference is neglected in this study, because the frequency is so low that it does not impose a significant effect on the service process. Besides, the cause of the interference is rather random, and can be solved by improving the management.

Besides, as the HTM are getting more vehicles, the depot Zichtenburg is required to handle more vehicles than before. The maintenance area is becoming inadequate and sometimes vehicles are also parked outside the garage. This lack of space if resulting in inefficiency and delays in some operations.

The third potential room for improvement is that the mechanics need to spend a lot of time on tasks that are in fact unrelated to their job, for example looking for material. This reduces the efficiency of mechanics. More clarification of job content and responsibility should be made to reduce the unnecessary trouble that people encounter in their work.

2.3 Key Performance Indicators

Everything is related to money. Money is actually the basic incentive for any company: improving the satisfaction of the customers can increase future income; together with reducing cost can improve the net profit of the company. Based on this rule and the impact that depots actually have on the system, two key performance indicators below have been chosen for the assessment in this project.

An important aspect that a transportation company interests in is the satisfaction of customers, because they are the eventual source of income. A robust system can provide a service that will not vary much when disturbances occur and can resume to its normal state within a short period of time. The term disturbance here is defined as refers to the disturbances that happen to the daily service provision. This means that the disturbances are short-term. Table 2.2 shows the number of vehicle changes that occur in the network per day. Type A changes are direct changes, where all travelers have to leave the vehicle directly and wait for the next vehicle to come. The problematic vehicle will drive to the nearest depot for repair. Type B changes are the ones in which vehicle will keep on driving until the end of this trip so that travelers can stay in the same vehicle.

Delays of service occur in daily operation and worsen when disturbances to the vehicles happen in the network. In transportation system, there are all kinds of disturbances, so nobody can ensure to the serivce provided to be totally on time. However, as travellers do not like delayed service, reducing average delay time can improve the quality of service and satisfaction of the travelers and avoid the potential penalty from

Vehicle Type	GTL		RR	
Change Type	Type A	Type B	Type A	Type B
Average	5.1	12.3	2.9	4.1
Standard deviation	2.8	3.5	1.7	2.1

Table 2.2: Number of vehicle changes per day

the municipality in the future.

There are different ways in defining delay of service. First, we can calculate the punctuality of the vehicles. If there is a service supposed to arrive at a certain stop at 1:00, but actually arrvies at 1:11, and another one at 1:10, but the vehicle actually arrives at 1:15, the delay of the vehicles are

$$(1:11-1:00)+(1:15-1:10)=0:16$$

This formula is easier to measure from the perspective of the transportation company, but it requires the fulfillment of all the service trips.

We can calculate the extra waiting time that travelers have to wait. In this case, the extra waiting time at the stop is

$$(1:11-1:00)+(1:11-1:10)=0:11$$

This formula starts from the perspective of the travelers, with comply better with the concern of customers' satisfaction. However, this method is more difficult to measure in reality. The busy stops, rush hours should also be differentiated to comply with this method, but it is too complicated and it does not encourage the fulfillment of service trips. In the end, the first method measuring the punctuality of the vehicle is chosen in this project.

Meanwhile, each kilometer traveled by the vehicle needs to be paid, because it consumes all kinds of resources. The deadhead-kilometer is even less desirable, because it is the distance that vehicles have to travel without providing service and thus does not bring the company any profit. The reduction of it can bring down the cost of the company. However, deadhead-kilometer is inevitable, because it is impossible to have a place to store vehicles at each end of the service lines. Besides, it is also economically undesirable to have too many depots because it will be costly to have the required equipement and human resource in each of them. To quantify the cost of distance that vehicles travel, the salary of the driver, amount of energy consumed, and the maintenance cost for mechanics and material sum up to 4 Euro per kilometer. According to this estimation and that the deadhead-kilometer is around 7% of the total distance travelled, the total cost of deadhead-kilometer is 3.7 million Euros per year.

Some other elements may be of minor concern in the definition of deadhead-kilometer, such as the distance traveled by both the problematic and substitute vehicle. This part is in fact a very small portion of the total deadhead-kilometer traveled by the entire network, because on average there are around 5 times of disturbances where vehicle change is needed for GTL vehicle, but there are more than 100 GTL vehicle driving more than 250 deadhead trips each day.

3 Methods for Studying Transportation Systems

3.1 Decisions on Different Levels

In companies, decisions are made on different levels. Strategic decisions usually involve the construction of new facilities (e.g., roads, railways, ports) and/or the acquisition of equipments and technologies [4, 5]. These decisions usually involve large capital investment. Due to the long lead time of these types of projects and long lifespan of the facilities, strategic decisions, once made, have large influences over other decisions that are made afterwards.

On tactical level, decisions usually concern issues of, for example, the design of work processes, the design of the timetable for travelers, management strategies, etc., which are easier to change than strategic decisions and have less influence. On operational levels, decisions normally concern the daily operations that all the employees in the company have to deal with, such as deciding the sequence of cleaning and maintenance for vehicles. Operational decisions are limited by tactical-level decision [5].

In order to solve the research problem in this project, a model with operational detail can be used. This is because such a model can be used for decisions on all the levels. It can be used to evaluate operational decisions, such as whether to dispatch a standby vehicle from one depot instead of the others. Collectively, the daily performance form the long-term performance, which can be used to assess the consequences of strategic decisions.

Decisions on different levels interact in such a way that the decisions from a upper level limit the decisions from the lower level. For example, once the infrastructure is built, it will last for decades, and it is very time-consuming and costly to change, the tactical decisions, such as design of service lines, made after are affected by the infrastructure, which is the result of a strategic decision. Similarly, tactical decisions will limit operational decisions, for example which vehicles can be cleaned depend on the design of the vehicle planning. These relationships are shown in figure 3.1.

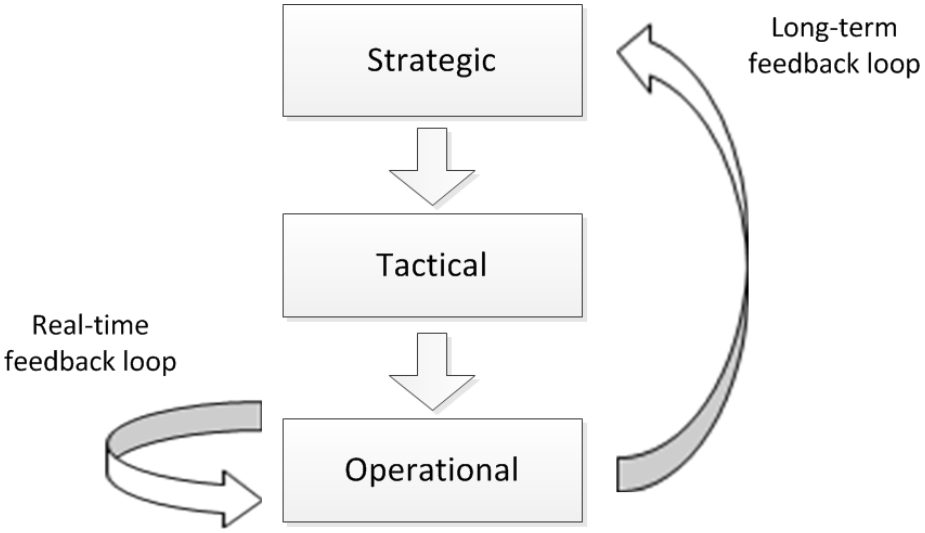


Figure 3.1: Levels of decisions and their relationships [6]

However, feedback loops also allow the decisions from a lower level to affect deci-

sions from the upper level with a window of months to years [6]. This is feedback loop is long-term because infrastructure of transportation system is usually costly and time-consuming to construct and difficult to change once built, and the it takes a long period to collect the operational data and to conclude an assessment on a strategic decision. Hence it is important to carefully assess the impact of the strategic decision before it is implemented. If it is possible to simulate and forecast the future impact of a strategic decision, the virtual 'infrastructure' can be much more easily built than that in reality, and this feedback loop can be shortened.

3.2 Research Methods for Transportation Network

Different methods have been used by researchers for the study of transportation systems. According to Petersen, models of transportation systems in general can be classified into three categories: (1) analytic (descriptive), (2) simulation, and (3) optimization [7]. The same classification can also be applied to the study of the depots. Below the different methods and their advantages and shortcoming under different situations are introduced.

3.2.1 Analytical and optimization models

Analytical models can be used to simulate the performance or behavior of the transportation system. They develop closed form or algebraic expressions for operating characteristics, track configuration or measurement of other attributes of the system that we are interested in [7]. By using values of the system which can be easily measured or controlled (such as service frequency, number of vehicles available), the models derive the measurement of performance or behavior of the system (such as possibility of delay of service).

For example, in a very simple case, by using the distances of from each depot to each end of service line and number of trips that vehicle have to go through this trip, the model can calculate total distance travelled of the entire network, which might be difficult to measure directly for the current situation and impossible to measure for a hypothetical system in the future. In this way, some detail or differences between the elements within the system are ignored, such as the detail movement of the vehicles. In the case of this study, in order to calculate the deadhead kilometers, more detail need to be included in the calculation. The design of the timetable, the number of vehicles for each line, the pattern of vehicle schedule (such as percentage of the vehicle are going back to the depots during non-peak hours) should all be included. Then, the final result can be calculated from all these parameters.

In order to calculate the delay of the service, stochastic models need to be used, because this is related with the uncertainty in the operations [8]. At the same time, many other characteristics of the depots need to be translated into mathematical parameters in the formulation. For example, whether the depot is only a parking place, an area for small maintenance, or full maintenance function area, can be translated as the amount of resources available in each location. Besides, there are also many uncertainties exist in vehicle disturbances and their impact on the performance of the system.

Optimization models also use algebraic expressions to characterize the system. In stead of calculating the indicators according to the given parameters, optimization models search in the solution space of some variables, trying to maximize or minimize a certain objective function for the interest of the problem owner.

For example in this case, in order to maximize profit or to minimize operational cost, an optimization model can search through the solution space, which contains combinations of locations and capacities of the depots. In this case, in order to calculate the objective function, all the status and behaviors of the system need to be expressed in mathematical formulations, such as the number of available vehicles in each depot, timetable of service, total distance travelled and delay of service [9].

Optimization models have had their application in transportation system as well, especially in the section of depot. This is because the choice of depot is a strategic decision, which influences the performance of the whole network, and it involves a long time horizon. Depot planning problem has also be associated with other objective, such as multiple-depot vehicle scheduling problem (MDVSP) and crew scheduling problem (CSP) [10] and optimal delivery route planning [11]. Many different problem classifications and algorithms have been involved in depot location planning [12, 13]. Among those algorithms, (Mixed) Integer Linear Programming has widely used, either directly by some general-purpose solver or (heuristically) by Lagrangian relaxation and heuristic methods [14].

For the study of transportation systems, analytical and optimization models have the advantage that they use pure algebraic formulations to characterize the system. In this way, the behavior and performance of the system can be understood in a more theoretical and generalized way.

However, as the number of elements of the system increases, the complexity of the analytical or optimization models increases greatly. Due to this reason, usually analytical or optimization models have their limitation on incorporating more details of the system. Besides, using stochastic models, the algorithms are normally limited by the type of distributions. With human intervention, which prevails in transportation systems, it is usually difficult to characterize with common types of distribution . This therefore limits the application of analytical and optimization models in this area.

3.2.2 Simulation models

Shannon defined that simulation is the process of designing a model of a real system and conducting experiments with the model for the specific purpose of experimentation, for the purpose of understanding the behavior of the system or of evaluating various strategies [15]. According to Brunner et al., transportation simulation has a long history as evidenced by numerous transportation simulation papers published. The simulation models have the advantage that more detail can be included, transient behavior can be modeled, and more data can be generated for analysis [7]. Many other paper have been devoted to the study of transportation system by the method of simulation models, especially discrete-event simulation models [9, 16].

Discrete-event simulation models have been used to simulate depots to provide a great understanding of the operations and strategies [17]. Many different aspects of the depots can be shown in the simulation, such as their function as a maintenance center, the internal working processes and the internal logistic design, and the rotation of vehicles between maintenance depot and normal emplacement [17, 16, 18]. Simulation models are flexible in their design, and can also be integrated with optimization models [19].

As we have discussed before, in HTM, the working processes within the depots do not affect the performance of service outside the depots on a significant level. Besides, it is more desirable to measure the performance of service instead of the efficiency or performance of the jobs inside the depot. Therefor, this project has been focusing on

the functionalities of the depots, such as storing vehicle and formulating contingency plan, instead of simulating the movements of the vehicles within the depot.

Sometimes the decision making process is simulated centralized in the models [19]. This central decision method is good when the processes is easier to control during the construction of the model because they are gathered in one place. However, in reality, decisions are in fact taken place in a decentralized term, such as each driver decide their own acceleration, halting time, or to wait for the vehicle in the front. Each object in the simulation is autonomous itself, and their individual behavior can emerge a pattern of the overall system which is difficult to be simulated by a central control mechanism. As a result, it is more appropriate to use to decentralized decision-making mechanism to simulate the vehicles' behavior in this project.

Verification methods of simulation models include aspects such as ensuring clarity of coding design, extensive debugging and adequate documentation have been used [17]. An important aspect of validation is that the client of the model has confidence in the model results. The level of confidence of the client can be increased through animation and walkthrough of the projects [17]. For this project, animation is available for LIBROS-II library, and walkthrough of the design can improve the understanding between the researcher and the problem owner.

Scenario analysis has been used for the uncertainties exist in the real system [20]. Common uncertain factors in transportation systems include different infrastructure, resource, timetable and technology.

Simulation model is suitable for this project due to several reasons. First, simulation models can be used to simulate behavior of the system on different levels, thus can support different types of decisions. Simulation models provide the decision makers a powerful tool to evaluate various strategies for the design and operation of the system. Simulation models can also provide the decision makers valuable insights into the behavior of the dynamic and stochastic system [19]. With the ability to incorporate great detail of the system, the model can simulate operational behaviors of the system, it can support operational decisions. This is done through the derivation of the status of the system for every time step, instead of trying to give a big description of the entire time period. Besides, the simulation model can incorporate much more information of the real system than other models and allow us to examine the system in much more detail. With further development, the simulation model can be used for many other study purposes. It provides a much broader ground for many different kinds of study, which can be always expanded and changed as the system evolves. With the information extracted from operational data and consideration of medium-term uncertainty. With consideration of long-term uncertainty and collective data generated, a simulation model can be used to support strategic decisions.

Another advantage of simulation models is that they can generate large of amount of data to perform all types of analysis, such as the movement of the vehicles, the formulation of the contingency plans, the delay time of each line in each hour, etc. They have the flexibility of customized output of data, which can be used for different kinds of research. In simulation models, not only single data of one specific movement can be collected, but also collective data, which can be used to show the performance of the entire system.

Third, simulation model can also be used for "what-if" test, to trace the consequence of one single change of the decision. In order to analyze the impact on deadhead-kilometer and robustness of network, we should not only analyze the impact on the final result, but also see how does this impact happen, increase understanding of the processes, and improve the system. Another great advantage of simulation is that

it is possible to conduct "what-if" analysis with complex system through simulation model, which will be very difficult otherwise with other types of methods.

Fourth, when dealing with a "dynamic" and "stochastic" system, simulation is a very power tool. The status of the system changes over time: the speed, location of the vehicles, the number of available vehicles in the depots and there are uncertainties involved in the system: the time and location of breakdowns of the vechiles, the length of the breakdown, etc. When there is variation in the processes of the system, such as arrival rates, internal design of the depots, design of vehicle schedule, it will be difficult to assess the performance by other means. Simulation models make it possible to simulate the variation of operation time and different people's interference in the process.

However, simulation models also have the disadvantage that details cannot be "ignored". For example, it order to evaluate a certain design of the depots, the detail vehicle schedule and vehicle allocations are also requirement to perform this assessment. If they are not available, it is impossible to proceed the evaluation until some assumptions are made.

Different tools have been used to generate discrete-event simulation models for transportation system. ARENA ™ is a popular one for discrete-event modeling. Its graphical interface is easy to use and it is process-oriented which is easy to understand and to present. However, it is difficult to use it to perform decentralized decision-making process and its simulation is based on processes. Some simulation tools and softwares for transportation system have also been introduced in some papers. For example, OpenTrack is one of the widely used software, which is a user-friendly and microscopic railroad network simulation program and has been mainly used to evaluate and test infrastructure plans and operating schedules to optimize network and timetable design. Users can define their own train, infrastructure and timetable, and can generate a large variety of data [21, 22]. LIBROS-II is an open source java library, which is used for rail operations simulation, which can use decentralized decision-making as java is object-oriented. Its advantage is that the programming library can give the user a greater flexibility in designing their other model. A more detail introduction of LIBROS-II model is given in section 4.2.

3.3 Decisions for the Future

3.3.1 Dealing with uncertainty

Due to the long time horizon of infrastructure, the long-term uncertainties need to be taken into consideration when studying strategic decisions of transportation system. There have been different methods for dealing with uncertainty defined by previous studies. For example, in Ebskamp's paper, he concluded several different kinds of methods for analyzing uncertainty, which are classified according to the different kinds of information the methods require and can provide. Among them, scenario analysis requires identification of key uncertain development and can give overview of how measures perform in different future [23, 24]. It is required for the design of transportation systems [4].

In the long-term, uncertainties can be external factors that are out of the reach of the problem owner. In a transportation system, the main driving forces can be economy, finance, technology and environment [25]. The combinations of the states of the driving forces can form different scenarios.

Scenario analysis is appropriate for the study in the project. This is because for a

long-term project, there are many different kinds of uncertainties, it is very difficult to know the distribution of possibilities of all the uncertainties. In addition, some factors are discretely distributed, such as the technology deployed, the service timetable, design of the routes of the lines. Besides, different scenarios can help the decision-maker to look into the future possibilities and design a robust solution instead of a "best" solution, which is in fact subject to all the assumptions and uncertainties. As a result, it is easier to give only some scenarios.

3.3.2 Multi-attribute utility theory

Multi-attribute utility theory has been used in policy analysis. In simplest form, multi-attribute analysis involves measurement of two basic components: weight and value. The overall utility of alternative x can be expressed as

$$u(x) = \sum W_{xi} S_{xi}$$

where W_{xi} is the weight of attribute i for alternative x and S_{xi} is the value of attribute i. A major challenge in multi-attribute analysis is to formulate appropriate methods for measuring weights and values [26].

Following the principles of multi-attribute value theory, there are different techniques to assess attribute weights, for example, analytic hierarchy process (AHP), direct point allocation, simple multi-attribute rating technique (SMART) [27], swing weighting, and tradeoff weighting.

Edwards explained that, arguments over policy are often about degree, not kind. Each time a decision has to be made, different parties with various preferences and interests fight over the degrees at a huge social cost. Multi-attribute utility measurement can spell out explicitly what the values of each participant, show how and how much they differ. With this method, explicit social policies can be defined with more efficiency and less ambiguity [27].

SMART technique was presented and first named by Edwards in 1977[27, 28]. The basic idea of this technique is that, every outcome of an action may have value on a number of different dimensions. The SMART technique is to discover those values, one dimension at a time, and then to aggregate them across dimensions using a suitable aggregation rule and weighting procedure [27].

According to Edwards, the SMART technique consists of ten steps: (1) identity the person or organization whose utilities are to be maximized; (2) identify the issue or issues (i.e., decisions) to which the utilities needed are relevant; (3) identify the entities to be evaluated; (4) identify the relevant dimensions of value for evaluation of the entities; (5) rank the dimensions in order of importance; (6) rate dimensions in importance, preserving ratios; (7) sum the importance weights, and divide each by the sum (normalize the weights); (8) measure the location of each entity being evaluated on each dimension (compute the score of each attribute to each alternative); (9) calculate utilities for entities (which use the equation above) and (10) decide [27].

SMART method also has the disadvantage that the spread of weights and the inconsistencies among the preference statements are dependent on the number of attributes present in comparison [29]. In fact, many researches have been carried out to compare the difference between weights under different conditions or by different techniques [29, 30, 31, 32]. As a result, the techniques of giving weights to the criteria, the conditions under which the technique is used, and the "bounded rationality" [33] of human beings all contribute in the uncertainty of weighting of criteria.

Compare to other techniques, SMART technique has been widely applied by its perceived ease of use [26]. It is based on extensive use of simple rating procedures, and can be easily taught to and used by a busy decision-maker. Moreover, it requires no judgment of preference or indifference among hypothetical entities [27]. Besides, since the number of criteria in this study is very small. There is no hierarchy or other complicated relationships among the criteria, so no need for complicated techniques to form the weights of the attributes, such as AHP. Besides, the weights given by the technique will be used only as a guidance, and the results of the evaluation will be tested against the uncertainty in the weights of criteria.

4 Discrete-Event Modeling: DEVS Formalism and LIBROS-II Library

As discussed before, Discrete-Event Modeling is chosen as the study method for the transportation system and LIBROS-II library is used as the programming tool. This chapter give an introduction to the structure of LIBROS-II library and how the model functions. This part of information is necessary to construct the model with LIBROS-II model, and will help the reader to understand the basic design principles in this model.

The underlying simulator of LIBROS-II library is DSOL, the Distributed Simulation Object Library [34, 35]. LIBROS-II depends on ESDEVS library [36], which is an recent development of the DSOL simulator implementing the parallel DEVS formalism [37].

4.1 DEVS Formalism

DEVS is the abbreviation of Discrete Event System Specification. It is a modular and hierarchical formalism for modeling and analyzing discrete event systems [38].

4.1.1 Atomic models

An atomic model has its own transition functions and can be expressed in the basic formalism of [39], which is expressed in the structure of DEVS as follow:

$$M = \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

where

X is the set of input values,
S is a set of states,
Y is the set of output values,
δ_{int} is the internal transition function for the state S,
δ_{ext} is the *external transition* function,
λ is the output function, and
ta is the set positive reals between 0 and ∞ [39].

The system is in some state s. If no external event occurs, the system will stay in this state for the resting time $ta(s)$. This time can be infinite, which means that the system is in a passive mode and will stay in this status forever if without external event. The transition time can also be 0, which means that the system will transit to next state immediately.

When the resting time expires, the system will transit to the next state according to the internal transition function $\delta_{int}(s)$ and stay in the new state s' for the new resting time $ta(s')$. And this process keeps going on.

If an external event occurs before the resting time expires, i.e., when elapsed time $e < ta(s)$, the system changes to a new state s' according to the external transition function $\delta_{ext}(s, e, x)$ and stays in the new state for time $ta(s')$. Then the same story continues [39].

In this way the all the transitions of states of the models are expressed by DEVS specification. And example with the transition functions of an atomic model, with code in java, is given in section 5.5.

4.1.2 Ports

Modeling is made easier with the introduction of ports [39]. Output ports can send value y, which is executed by the output function λ. Input ports can receive values x from outside, which will trigger the external transition function δ_{ext}.

4.1.3 Coupled model

Coupled models are expressed by using the coupled model specification – essentially providing component and coupling information [39]. A coupled model does not have its own transition functions. Its state depends on the atomic models that are internally coupled to it.

Coupling is, in fact, to make connection between the ports of the models. The coupling relations are illustrated in the figure 4.1. When a model has internal couplings with another model, it means that the output port of this model is connected with the input port of the other model, so that the output of this model can be transmitted to the other model. A is internally coupled with B, A.*out* $\rightarrow$ B.*in*. If a model has an external input coupling with another model, then the message through this input port will be transmitted to the input port of the child model. Here AB is the parent model, and A and B are the child models, AB.*in* $\rightarrow$ A.*in*. If the model has external output coupling with another model, then the message going through the output port of the child model will be sent to the output port of this model, B.*out* $\rightarrow$ AB.*out*. Internal and external couplings are used as message transmission channels.

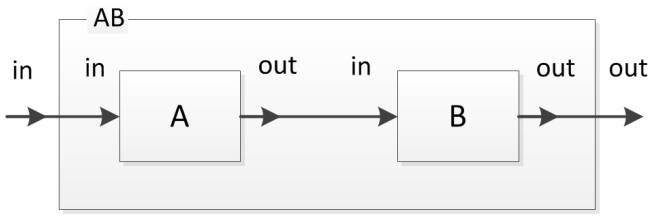


Figure 4.1: Coupling in DEVS [40]

4.2 LIBROS-II Library

LIBROS-II is an open source java Library for Rail Operations Simulation [37]. The model detail of how some important classes are implemented in LIBROS-II model is introduced.

4.2.1 `AtomicModel` and `CoupledModel`

In LIBROS-II the transition functions of the atomic model in DEVS are defined in class `AtomicModel`. Method `deltaInternal` defines the internal transition functions, `deltaExternal` defines the external transition functions and `lambda` defines the output function. Variable `sigma` specifies the time advance of the next internal transition, which is the resting time of the model in the current state. When an internal event occurs, *lambda* is first executed to calculate the output of the model, and

`deltaInternal` calculates the next state of the model. When an external event occurs, `deltaExternal` is executed. Then `lambda` and `deltaInternal` are used to compute the output and the next state of the model.

The internal coupling method is executed via the parent of both models. So, the models to be internally coupled need to be on the same hierarchical level, which means that they have the same parent model.

Messages are propagated through `InputPort` and `OutputPort` in LIBROS-II model. Each of them is dedicated to one type of message. A `StartNode` has one input port for message forward and one output port for message backward. A `EndNode` has one input port for message backward and one output port for message forward. Through coupling among the models, nodes are connected to each other, which builds the channel for message propagation.

4.2.2 `RailVehicle`

One of the most important part in the model is to simulate the movement of the vehicles in the network. The vehicles drive out of the depots, travel through intersections, stop at halting places (stops), provide service trips on difference directions of the line, and drive back to the depots.

Rail vehicle (`RailVehicle`) is an atomic model. The movement of the vehicle is guided by the shape of track segment, the direction of the switches, the signals of the control units, the speed limit of the infrastructures, etc. According to this information, the `RailVehicle` updates its states when the environment or the resting time of its current state expires.

When moving along the tracks, every time when the rail vehicle changes it state, it sends messages back and forth. From the messages that it receives, it can obtain the information such as the distance and speed limit of next infrastructure, the distance and speed of the vehicle in front. According to this information, the vehicle decides if it needs to accelerate, decelerate or move with the current speed. When a vehicle travels on a track segment, it is internally coupled with this track segment, so that the messages can be propagated through the track segment. When the vehicle is traveling at a `Stop` or a `Interlocking`, it is internally coupled with this blocks. As a result, through finding its parent model, it can communicate with these blocks and fetch information such as the name and the service timetable of the stop. More detail about `RailVehicle` please refer to Appendix B.5.

4.2.3 `Sink` and `Source`

The following models are components of the track network on which vehicles move.

Class `Sink` is an atomic model. When a sink receives a message and the sender of the message is connect to the sink, the sink will delete the sender (a vehicle) from the network. In the original model, sinks are place after the last stop of the line, so that the vehicles will be deleted after finishing one service trip.

Class `Source` is a coupled model. It contains at least one `VehicleGenerator` and a track segment which is directly connected to this exit of the source. The vehicles are generated by the `VehicleGenerator`, and "connected" to the track segment of the source, and then they "drive" out of the `Source`.

Class `VehicleGenerator` is an atomic model. A `VehicleGenerator` stores the information of the timetable according to which the vehicles are "created" in the

model. The information stored in the vehicle is also initialized by the VehicleGenerator. The class VehicleGenerator does not have external transition function, so its internal variables cannot be changed after creation. More detail about VehicleGenerator is in section B.6.

4.2.4 TrackSegment

Each track segment (TrackSegment) has one shape and one speed limit. It has one start node (StartNode) and one end node (EndNode)a type of SimpleInfraElement. The start node of a track segment is connected to the end node of the infrastructure in front of it along the direction of the rail road. In this way, the infrastructure can propagage message along itself, which is used to calculate the distance of the vehicle ahead, the distance of next infrastructure, and to send information of infrastructure, such as speed limit and traffic signal. A TrackSegment can be a component of a Source, a StopBlock, or an Interlocking.

4.2.5 Interlocking

An Interlocking simulates an intersection in the network, where more than one piece of track are joint together and switches are used to guide the directions of different vehicles. An Interlocking model stores the information of routs for different vehicles. When a vehicle is approaching a cross, the interlocking receives the message from the vehicle, which includes the identity of the vehicle that could be composed of the line number of the vehicle. According to this information, the interlocking finds the route for the vehicle and change the direction of switch accordingly. In case that multiple vehicles are approaching the same cross, the Interlocking will decide the sequence of vehicles passing through depending on the priority of vehicles. The other vehicles who can only pass after another vehicle can only wait in front of the cross until the interlocking gives it permission to pass.

4.2.6 NodeMessage

Messages carry values that are sent through coupled ports. If an object changes its state, e.g. a control signal turned from green to red or a preceding vehicle reduced its speed, the object notifies the approaching vehicle using the publish-subscribe (also called event notification) interaction scheme [41].

Infrastructures (including tracks, stops, traffic lights) and vehicles use message which can travel via message ports between components connected to each other. Much information can be gained through the sending of messages, such as the number of line the vehicle is serving, distance of vehicle in front and order from traffic lights.

4.2.7 CadModelBuilder

The simulation model is mainly built and initialized in this class. It is a sub-class of the TopLevelModel. Since DEVS is a hierarchical formalism, when generated, each model has a parent model, except the TopLevelModel which is on the highest hierarchy of the model.

1. The model reads the AutoCAD file in a way that it starts from the the blocks called “source”. From the “source”, the model can find the track that is connected to it, and then the next entity that is connected to this track. This procedure goes

on until there is no more entity connected to the current one any more. The model goes through all the entities that are connected in this way, and put all of blocks, including "source", "stop" and "point", in the lists of each type. And the lines, which represents tracks in the network, are also put into a list called *trackEntities*.

2. Every "source" should have only one track connected to it. The direction from the source to the track defines the direction that the vehicles can travel along the tracks. The tracks in the following will define the direction accordingly. An *infraEntityMap* is created to store the information of each pair of connected track segments.

3. The model uses the list of the "point"s, together with the track segments connected to them, to generation `Interlocking`s. The model then uses the list of "stop"s, together with the track segments connected to them, to generation `Stop`s in the model. After this, the model tree is built, with the elements of the transportation system, including `Stop`s, `Interlocking`s, and `TrackSegment`s.

4. The model then starts to build the network. From each "source" from the list and its information from the AutoCAD file, the model generates a `Source`. The model looks for the next stops that the directly connected to this source, which are the starting points of the lines. Then the model uses the service timetable of these stops to estimate the time for generating vehicles, creates a `VehicleGenerator` for each `Timetable` of generating vehicles, and adds it to the source.

5. Afterwards, the model will look for the next entity connected to this source through the relationship stored in the *infraEntityMap*. The timetable of the `Stop`s and the defined routes of the `Interlocking`s are loaded from the input Excel files. When there is no more entity connected to the current one, the model will automatically generate a `Sink` and connect it to the last track segment.

5 Design of the Model

The design of the analyzing tool is an important part in the project. In this chapter, the requirements of the model, the conceptual design and the detailed design procedures are explained.

5.1 Requirements of the Model

In section 2.3, I have discussed the key performance indicators that are important in evaluating the transportation system in this study. In order to measure those indicators, the model should have the following functionalities:

1. Vehicles travel over all the distance in the network as they should do in reality perform.
2. Disturbances can happen randomly in the network, in which case a vehicle will break down.
3. A central controlling mechanism can receive the information of disturbance, formulate contingency plan and dispatch a standby vehicle to resume the service.
4. The vehicles can differentiate deadhead trip and normal service trip and record the distance. Delays of service can be measured and recorded.

5.2 Conceptual Design

In order to fulfill the requirements mentioned above, a more detailed conceptual design of the model is introduced below, and figure 5.1 presents the important parameter of the classes and the relationships among them.

1. The movements of the vehicles over the entire network should be simulated, which include traveling between the depots and the ends of the lines and traveling along the service trips. As a result, the vehicles should appear at the exits of the depots and disappear at the entrances.
2. `Exits` of the depots are where the rail vehicle (`RailVehicle`) "appear" in the track network of the model, and `Entrances` of the depots are where the vehicles "disappear" from the track network. When a vehicle leaves or enters a depot, the information of the depot will be updated, for example, the number of vehicles or of standby vehicles. All these entrances and exits of the depots are connected and share the same information through the coupled model `Depot`. Every depot consists of all the entrances going into it, all the exits going from it, and a depot control unit (`DepotControlUnit`). Information such as number of available standby vehicles can be fetched and updated from any entrance or exit that belongs to this depot. With the order of the control center, the depot can dispatch an extra vehicle, which was not included in the initial timetable for dispatch.
3. Vehicles store their own vehicle schedule, which contains the information of all of their service trips. In this way, after finishing one service trip, the vehicle knows where to go next: go back to the depot or start the next service trip on the other direction. If it is going to start the next service trip, it will wait at the first stop till the service time if it arrives too early.

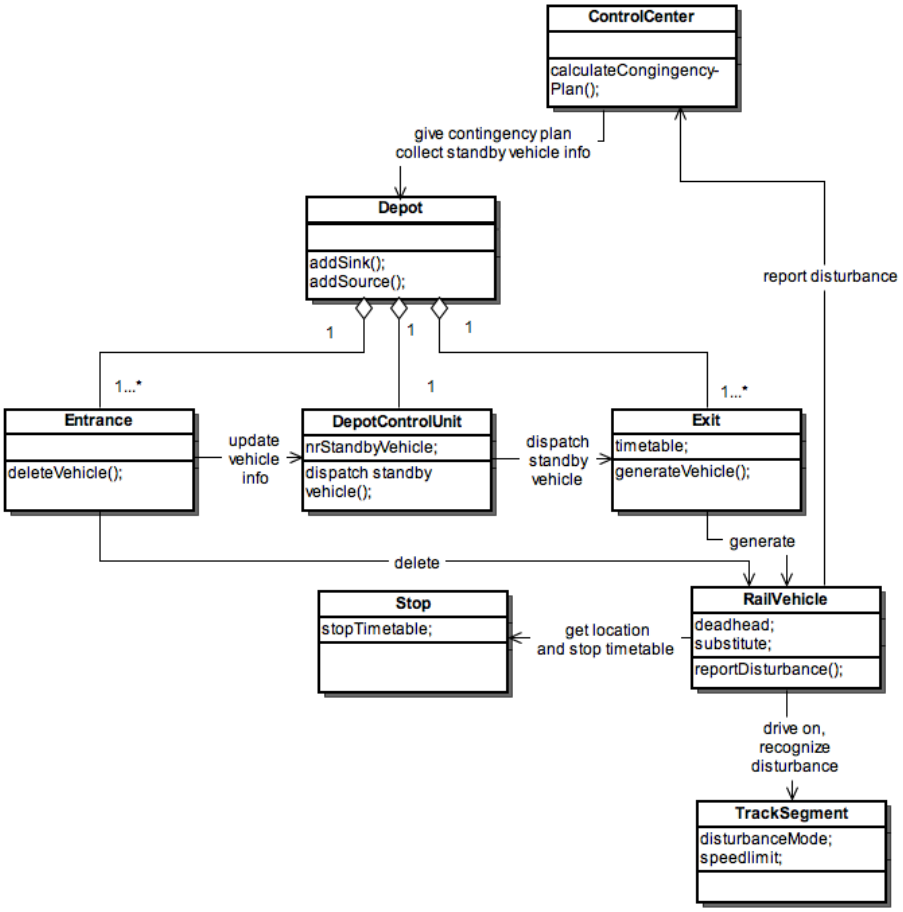


Figure 5.1: Conceptual design of the model

4. When a vehicle arrives at a stop (`Stop`), the information from the stop can help it recognize its position in the network. The vehicle can know whether it is at the start or the end of the line, or the place where it should resume the service if it is a substitute vehicle. Through comparison with the service timetable from the stop, the vehicle can also know if it is early or late for the current service trip of the stop.

5. The locations of vehicle disturbances are decided through choosing a random track segment in the network. When a vehicle comes in front of the track segment, it knows that it "should" break down and stop in front of that track segment.

6. Vehicles can report the disturbances to the control center (`ControlCenter`). In this case, control center can communicate with the depots to collect information of the number of available standby vehicles in each depot. The control

center then formulates and sends contingency plan to the chosen depot to dispatch a standby vehicle to resume the service.

5.3 Specification of Key Classes

Some key classes are important to realize the conceptual model mentioned above. This section introduces the structures and functions of these key classes individually.

5.3.1 `Depot`

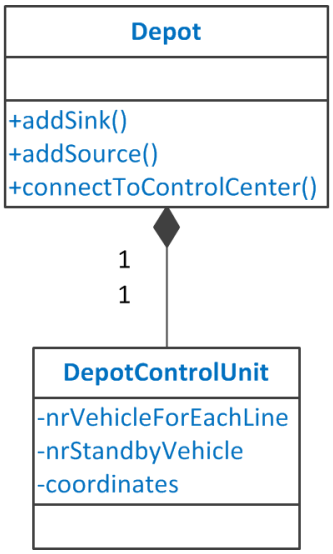


Figure 5.2: Class diagram: `Depot`

Class `Depot` is a coupled model. It is a type of `InfraComponent` (section B.2, which means that a `Depot` can have more than one start node (`StartNode`) and more than one end node (`EndNote`). It contains one `DepotControlUnit`, at least one `Entrance` and at least one `Exit`, as shown in figure 5.3. Figure 5.4 gives an example of the depot Zichtenburg. From the figure we can see, there are different entrances and exits of the depot, locating at the top and the bottom of the figure. In order to let all these entrances and exits be able to share the information and communicate with each other, they need to be all included in the model `Depot`. By giving attributes with the name of the depot and numbering as suffix to the blocks of "source" and "sink" in the AutoCAD file, the model can recognize them and couple them with the same `Depot`. Example of entrances and exits belong to the same depot is shown in figure C.1 and C.2. See figure B.2.

The details of couplings of a `Depot` are shown in figure 5.5. All the depots are coupled with the control center to exchange message through *outputPortToControlCenter* and *inputPortFromControlCenter*. All the `Entrances` have input external couplings with the `Depot` so the messages and vehicles can enter, all the `Exits` have output external couplings with the `Depot` so that they can send messages and vehicles. All of the `Entrances` and `Exits` have internal couplings with the `DepotControlUnit` to exchange and update information.

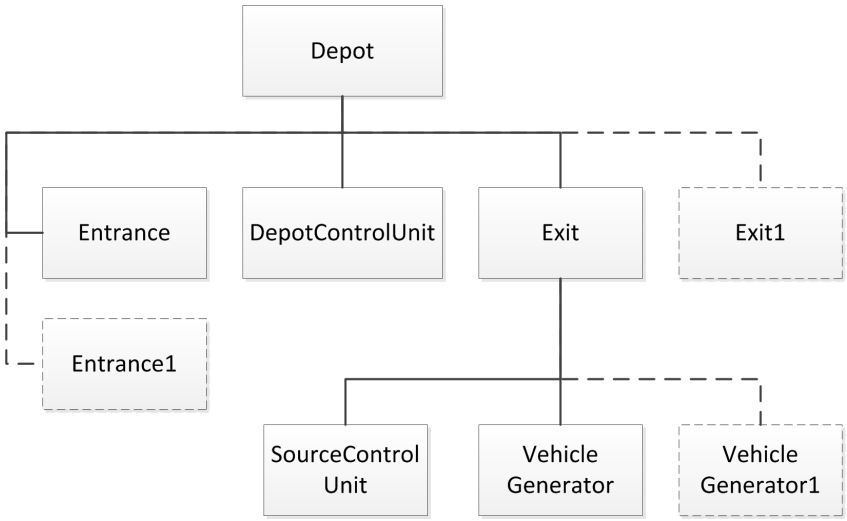


Figure 5.3: Structure of `Depot`

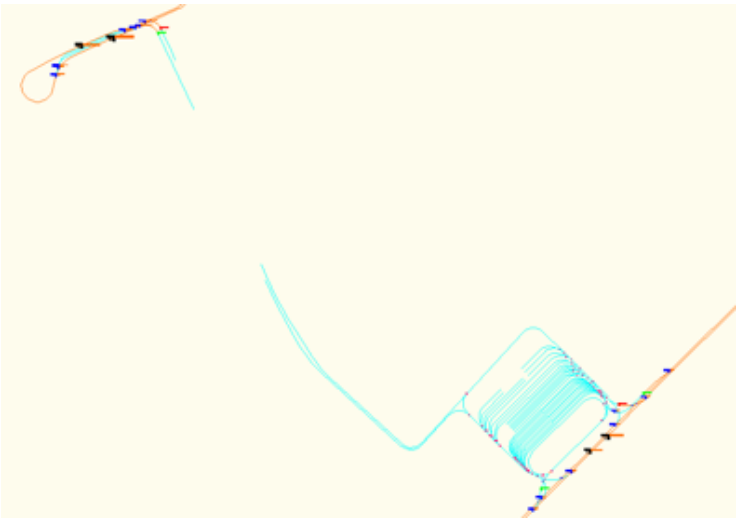


Figure 5.4: Depot Zichtenburg

A `DepotControlUnit` is an atomic model. The *DepotControlUnit* functions as the central controlling part of the depot. It also keeps certain information of the depot, such as location of depot in terms of coordinates and the number of available standby vehicles in the depot.

Figure 5.6 shows the state diagram of `DepotControlUnit`. For the the state diagrams, triangles on the left side represent the input ports of the model and those on the right side represent the output ports of the model. In the middle, the eclipses with solid line represent passive states of the model, whose resting time is infinite, and those with dashed line represent transitionary states, whose resting time is a positive value. Solid arrows represent external transitions, next to which the active port is shown above the "?" and the expected value from the message is shown beneath it. While dashed

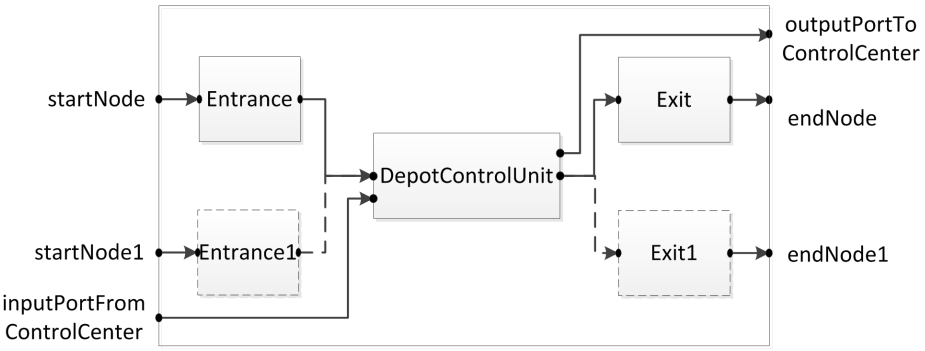


Figure 5.5: Nodes and ports of communication in `Depot`

arrows represent internal transitions, next to which the output port is shown above the "!" and the output value is shown below.

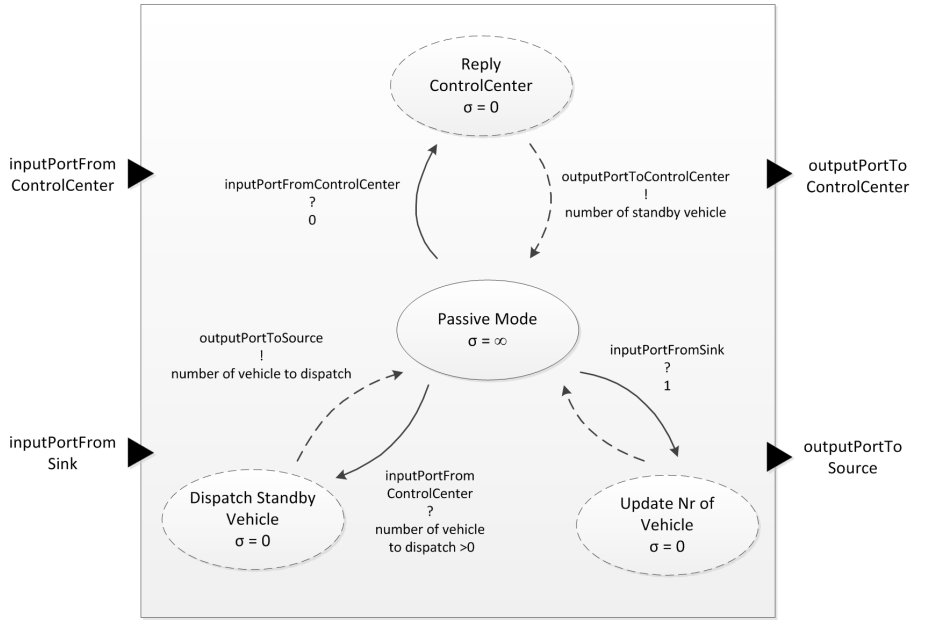


Figure 5.6: State Diagram: `DepotControlUnit`

In normal situation, the depot control unit is in a passive mode. When it receives a message from the `Entrance`, it means that a vehicle has "come back" to the depot. Consequently, the depot control unit will update its information on number of vehicles inside the depot. After this, it returns to the passive mode. When the depot receives a message from the control center, it will transmit the message to the *inputPortFrom-ControlCenter* of the depot control unit. If the message is to request the information of number of available standby vehicles, prepares the message, sends it back to the control center through *outputPortToControlCenter* immediately, then turns back to the passive mode. When it receives an order from the control center to dispatch a standby vehicle, it will send a message to the `Exit` of the depot, with all the vehicle information that

comes with the message for this standby vehicle to provide the service. After sending this message, the depot control unit transits back to the passive mode again. See figure B.3 and B.4.

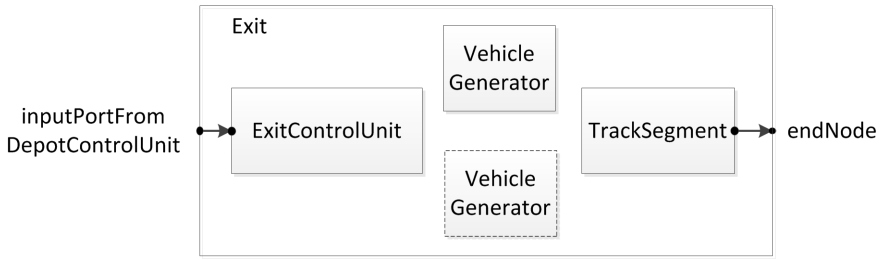


Figure 5.7: Structure and coupling of class `Exit`

Figure 5.7 shows the structure and coupling of the class `Exit`. Class `Exit` is a subclass of `Source`, which means that the `Exit` is a coupled model. Therefore its states and actions depend on the states and actions of its child model. In order to make the `Exit` be able to react to the order of dispatching an extra vehicles in case of accidents, `ExitControlUnit` is used. A `Exit` contains an `ExitControlUnit`, a *inputFromDCU* (connected to the depot control unit), schedule to dispatch vehicles, initial values of the number of vehicles stored for each line, and a list of the lines that this depot will dispatch vehicles for. The `ExitControlUnit` is an atomic model. Its input port is externally coupled with the input port of the parent model `Exit`. When the `Exit` receives a message to dispatch a standby vehicle (which is from the depot control unit), it sends the message to the `ExitControlUnit`. As talked about before, the timetable inthe `VehicleGenerator` cannot be externally changed once created. In order to generate a new vehicle, then the `ExitControlUnit` finds its parent model (the `Exit`), makes the parent model create another `VehicleGenerator`. See figure B.5.

5.3.2 `ControlCenter`

The `ControlCenter` is an atomic model. It has input and output ports to communicate with depots. It has another input port to receive disturbance report from the vehicles.

Figure 5.8 shows the state transitions of the control center. For disturbance, the control center is centrally controlling the network and can communicate with depots and vehicles. A random number is used to decide whether the disturbance will happen in this hour. If the random number falls in the region that disturbance does not happen, the control center waits for a certain period of time (1 hour here) and gets the next random number. If the random number falls under the possibility of disturbance, which means that a disturbance will occur in this hour, the control center transits into the state “DisturbanceStart”. In this state, the control center “choses” randomly a track segment and “sets” its speed limit to 0 and in disturbance mode. Afterwards, the control center changes into a passive mode “WaitForReport”. When receiving a report from the vehicle, the control center changes to state “CollectStandbyVehicleInfo”, firstly sends messages requesting number of vehicles in each depot, and then turns into a passive mode again while waiting for the reply.

In order to make sure that all the message are received before any action is taken, a

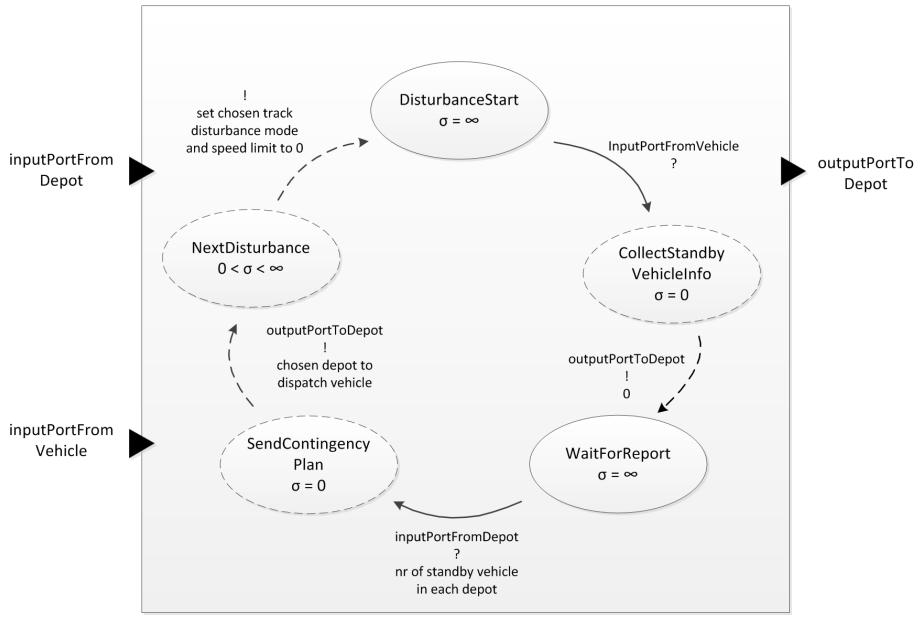


Figure 5.8: State Diagram: ControlCenter

0.1 second a time advance is set here. As 0.1 second is small enough that no significant event will happen during the time and we can make sure that all the relies have arrived without counting them.

After this, it reaches state "SendContingencyPlan", and formulates a contingency plan, and sends this plan and the information of the disturbed vehicle to the depots. Because in the model, the simulation of sending messages and replies are all instant. In the end, the control center comes back to the state "NextDisturbance" and the old story continues. The code of the `ControlCenter` is explained in section 5.5.

5.3.3 `RailVehicle`

Figure 5.9 shows the state diagram of class `RailVehicle`. In LIBROS-II, these states do not exist, and the rail vehicles are actually going through internal and external transitions and changing their states constantly. For the sake of simplicity, it is here assumed that there is phase "Normal running in service" for the rail vehicle. For rail vehicle, after getting out of the depot, it is normally driving on the track according to the directions directed by the `Interlocking`s. If the vehicle is a substitute for another vehicle, which means that it is dispatched to resume the service left by a disturbed vehicle, then after arriving at the last stop before disturbance, it will turn into a normal vehicle running on the track. When a disturbannce happens to a vehicle (in this case a breakdown of the vehicle), the vehicle will report disturbance with its own service information to the control center and then remove itself from the model. The process is simplified here and because the rest of the distance traveled by the problematic vehicle is of little concern of the study (section sec:Indicator.

After the vehicle leaves the depot or the last stop of its service trip, it is in the state of "Deadhead trip", which means that it will accumulate the deadhead distance it travels. When it comes to the beginning of the service line, its stops accumulating its deadhead-

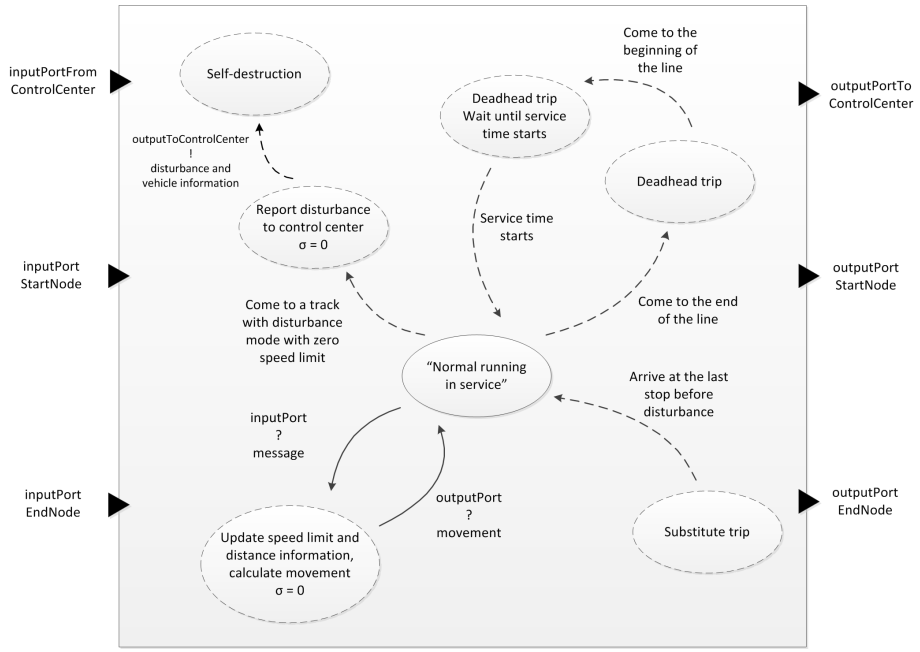


Figure 5.9: State Diagram: RailVehicle

kilometer, waits for the service to start ("Deadhead trip Wait until service time starts") and then goes back to the "normal running" state when the service time comes. When a vehicle arrives at an area which is blocked because of a vehicle disturbance before, it will wait in front of the area. Till the vehicle is informed that the disturbance is finished, it will reset some of its attributes and start driving on the track.

In real life, the drivers can easily tell where the vehicles are and the external situations to perform all those transitions of states mentioned above. However in the simulation model, things are not so easy as they appear in real life. In order to realize those functions, several attributes have been used in `RailVehicle`, as shown in figure 5.10.

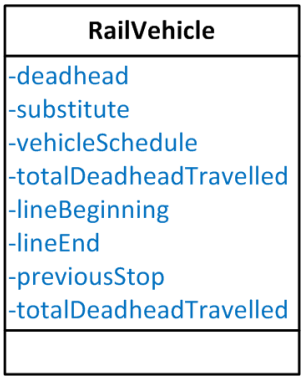


Figure 5.10: Class Diagram: `RailVehicle`

When a vehicle approaches an intersection, it sends certain information so that the

`Interlocking` can use this information to find out the route for the vehicle. This information is defined by both the line number and the destination of the vehicle. If the vehicle is traveling from the depot or the end of the line to the starting point of its next trip, it will send both its line number and the name of the first stop. If the vehicle is traveling to to the depot, it will send both its line number and the name of the depot. If the vehicle is a substitute, which means that it is going to the area of the disturbance to resume the service, it will send "0" in the place of the line number plus the name of the destination stop. Here its line number is defined as "0", because now the vehicle does not need to follow the normal deadhead route of a certain line, which can lead it to the beginning of that line. And destination is defined according to the last stop that the vehicle passed before broke down, which is stored in attribute *previousStop*. A new method *getVehicleIDForInterlocking()* is used to define this information. This information is also used in class `FacingPoint`, thus should be changed accordingly as well. See figure B.6.

Every time when a vehicle arrives at the first stop of the line, if the vehicle is not a substitute vehicle, the vehicle sets *deadhead* to be false. If the stop is at the end of the line, and the vehicle is not a substitute, the vehicle sets *deadhead* to be true, updates the name of stops of the begining and end of next service trip to prepare for the next trip. Sometimes in order to reach the beginning stop of the line, the vehicle has to pass the end stop of the other direction. Normally they have the same stop name, but service timetables. In order to avoid miscalculation of the delay time, the name of the names of the end stop of the line is added with an "e" in the AutoCAD file, as in figure C.3. See figure B.8.

Some data of should only be updated when the vehicles have just arrived at the stops, for example the delay time. However, as `RailVehicle` goes through its internal transition every time it finishes one movement, the data could be updated for several times within the same stop, and the final result could be very different from the initial one if the vehicles stay at the stop for a long time for some reason (such as the vehicle waits at the beginning stop of the line until the service time begins, or the vehicles have to queue in the stop because of halting of the vehicles in the next stop which is very close). Attribute *previousStop* keeps track of the name of the stop when the attribute is updated last time, so it can make sure that those attributes are only updated once for each stop. When a vehicle passes by a stop, a *tripID* for the stop is updated to get the corresponding service time for this trip. The *tripID* for the stop is updated with updating *previousStop*, which makes sure that every time a vehicle passing by only gets one "service time". Then the delay time is computed as the difference between the arriving time of the vehicle and the service timetable from the stop for this service trip. One exception here is that if the vehicle arrives early at the first stop of the line, it will update the delay as 0 in stead of a negative number. This is because that the vehicle will wait at the stop and leave when the service time starts, so this vehicle should be "perfectly" on time. See figure B.7.

In the transition function of rail vehicle, there are a few places where the *totalDistanceTraveled* is updated. In order to keep record of the deadhead-kilometer traveled, the attribute *deadheadTraveled* is also updated at the same place if attribute *deadhead* is true. See figure B.9.

A rail vehicle can have several movements within the stop block. Every time the rail vehicle has a new movement, it goes through all the internal transition functions. The dwelling time is used as the resting time of the `HALTING` phase, which will only happen when the vehicle has completely stopped at the end of the stop block. As a result, we want the extra dwelling time to be calculated right before `HALTING` phase

starts, not when the vehicle is still queuing behind another vehicle in the stop block. As a result, dwell time is calculated right before the phase of the vehicle changes from `STOP` to `HALTING` in a stop block. See figure B.10. When updating the delay time, the it has to be made sure that the vehicle is on a service trip, which means both *deadhead* and *substitute* are false. If it has arrived too early, it will calculate an extra dwelling time, which will be added to the dwelling time of the vehicle at this stop. See figure B.11.

However, when we can calculating vehicles' delay time, we want to use the time when the vehicle first arrives at the stop block, not after the vehicle has been waiting for a long time for a vehicle in the front to start. In order to realize this, *previousStop* is used here. When the current stop name is different from *previousStop*, it means that this vehicle has just arrives at the stop block, and the delay time is computed and *previousStop* is updated as the current stop name. If the stop name is the same as *previousStop*, it means that the vehicle has arrived at this stop block some time ago, the computation of delay time is not necessary anymore. This *previousStop* is also used to show where to resume the service after a disturbance occurs. When calculating delays, there are many things should be paid attention to. For example, normally the delay time is calculated according to the time when the vehicle first arrives at the stop. At the first one stop of the line, the vehicle may arrive 5 minute earlier and wait there until the start time of the service. It would be unfair to include this 5 minute in the delay time, which will be calculated as "-5" minute of delay. As a result, in the model, if the stop is the first one of the line and the delay time is negative (which means that the vehicle arrives earlier), this delay time will be recorded as 0.

Current the substitue vehicles drive to the last stop which the disturbance vehicle passed by. The way that HTM is doing it is actually more flexible than this. Because for the busy lines, such as line 1, the interval between consecutive vehicles are around 10 minutes normally. Unless the location of disturbance is very close to a depot, ususally it would be difficult for the standby vehicle to arrive before the next normal service vehicle. As a result, if we let the standby vehicle to drive directly to the location of disturbance, it does not usually hlep to reduce the time that passengers wait at the stops after the disturbance. It simply makes the calculation of delay of vehicles easier, as every vehicle will have one corresponding service time to measure their delay time. This also makes sense when we are using punctuality of the vehilcle as a measurement.

5.3.4 `TrackSegment`

The track segment (`TrackSegment`) here also supports some functionalities for the "disturbance". Its state diagram is shown in 5.11. The state of the track segment is in fact controled by the control center. A track segment is normally in a passive state. When it receives a message from input port, it propogates the message through the output port and returns to passive state. When a track segment is chosen to be the area of disturbance, it turns into another passive state with disturbance mode and a speed limit of zero. After a vehicle "breaks down" at the area, the control center makes it leave the disturbance mode, but still remaining a speed limit of zero so that the road is "blocked" for some time due to the disturbance. After the time of disturbance finishes, the track will be set back to its original speed limit and remain in the passive state.

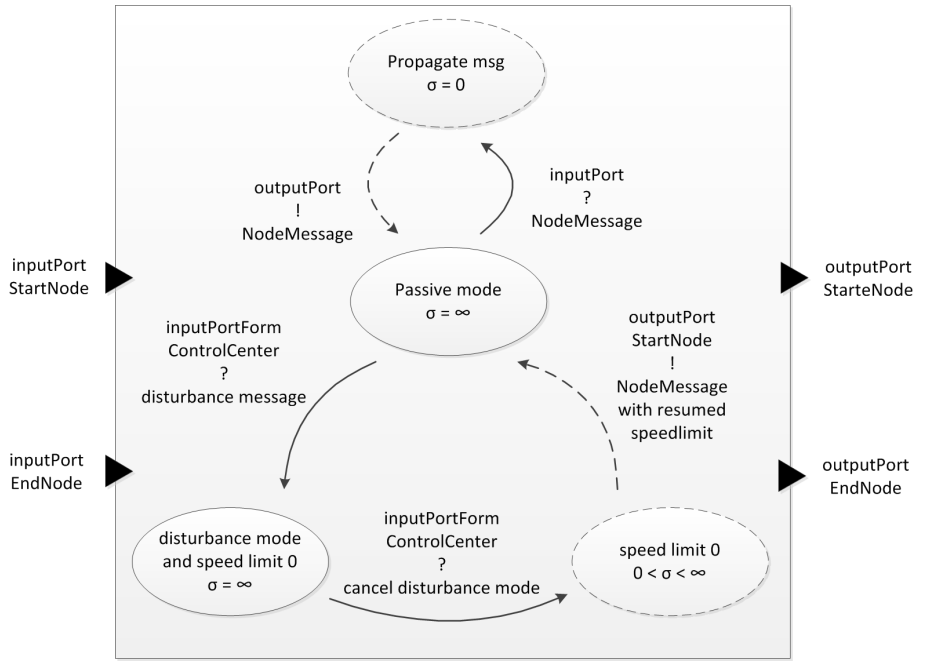


Figure 5.11: State Diagram: TrackSegment

5.4 Detailed Design

5.4.1 Assumptions and simplifications

- Five lines operated with the GTL vehicles are taken into the analysis. RR vehicles served lines are not yet finished in the model, due to extra construction work of the safety system.
- There are two types of vehicle changes mentioned in section 2.3. Only type A changes are considered, because then vehicle changes are required.
- Disturbances can only occur in certain places in the network. This is because currently there is no method for the vehicles to find their way to a certain place automatically; all the routes need to be defined manually.
- Vehicles do not return to the depots during the day. This will result in a lower measurement of deadhead-kilometer in the end. However, because for each line, the percentage of vehicles going back to the depots during the day is the same, the final results are still comparable among different lines and among different alternatives.

5.4.2 Preparation of input files

This part introduces how to prepare the Excel input files, which contains most of the input that the simulation model needs.

All of the information specific to the network is imported through the input files. The input of the optimization is changed. originally is the ideal number of vehicles, starting from each direction of the trip.

Service Timetable Sometimes the stop ID specified in the AutoCAD file does not correspond with those in the timetable. This mostly happen to the two stops with the same name but on the opposite sides. Because the service timetable is loaded according to the stop ID, so this will result in incorrect computation result. Examples are stop Central Station with stop ID 2603 and 2603 for line 17. This part is corrected in the timetable file.

Vehicle schedule in order to construct vehicle schedule from service timetable, the following steps are followed. The numbers of vehicles starting from each direction are taken as defined by the company. The vehicles are dispatched for the first service trips from each direction. Assum that vehicles do not go back during the day. According to the ending time of the trips in the service timetable, the rule of "first come first go", and the estimated time it takes for the vehicle to turn at the end of the line, the service trips from the other direction are also assigned to the vehicles. The vehicles keep circulating in the line untill all the service trips have been assigned to the vehicles. In this way, all the service trips are assigned to a "sequence number", which corresponds to one of the vehicles from the line.

If some service trips cannot be served in time according to this rule, then the number of vehicles assigned for the direction should be adjusted. For example foe line 1, the input of the optimization solver is that 6 vehicles starts from Scheveningen and 8 vehicles start from Delft Tanthof. However, due to the fact that in the current service schedule, the service from Scheveningen starts one hour earlier than that from Delft Tanthof. So after those 6 vehicles have all been dispatched for service, the vehicles from Tanthof will not have arrived yet. As a result, if we follow completely the result from the optimization solver, some trips will not be served or be delayed due to that the vehilces starting from the other end of the line cannot arrive on time. Adjustment need to be made. In order to wait for the first vehicle from Delft Tanthof to arrive, there need to be at least 10 vehicles starting from Scheveningen. So in order to keep the total number of vehicles for each line, we put the constrain of number of vehicles starting from Delft Tanfhof 4.

Optimization solver After the numbers of vehicles for each direction of the lines are decided, they are put in the optimization solver, it gives an solution of to which depot each vehicle belongs to, with an objective of minimizing of the total distance of all the vehicles traveling between the depots and the ends of lines. The distances from each depot to each end of the line and the capacity of the depots are used as constraints of the optimization solver.

Vehicle routing As currently, there is no better way of specifying the routes the vehicles travel, all the routs need to be added to the "services" and "deadheadRoute" file. In the "deadheadRoute", every route takes two columns. In the first column on the first row, the line number needs to be specified. In case that this trip is for stand-by vehicle to substitute a broken one, "0" is used here. The name of the exit from which the vehicle is dispatched is specified in the second row. Below, the code for all the interlockings that the vehicles will pass by along the way are listed, and the direction of the switch is specified in cell on the right side of the code.

Initial value The initial values of the depots are specified in Excel file "initialValue". This file takes the output of vehicle allocation among the depots from the optimization

solvers. For each `Exit`, the line number plus the name of the first stop is listed in the first row, and the number of vehicles of this direction of the line allocated to this depot is listed in the row below. At the end of the row, the number of stand-by vehicles is listed under the cell written "nrStandbyVehicle".

5.4.3 Preparation of input AutoCAD file

- Create a new block in AutoCAD file called "sink", with an attribute with the name of depot that it belongs to. Put those blocks at the ends of the tracks from where vehicles go into the depots.
- More tracks and switches leading to the depots need to be built. New information about how switches should be changed for each line should be added to the Excel file *services.xls*. Some tracks are already in the AutoCAD file, but are in a different color, which are not read by the model. They can be changed to readable by chaning the color in AutoCAD file.
- There are certain rules need to be followed when constructing the tracks in AutoCAD file. For example, between a exit and any other block, there needs to be at least two separate tracks. This is because the `Exit` class in the model contains one track coming out from the exit. Besides, other blocks such as "stop" and "point", which represent a halt in the line and an interlocking where two tracks meet, are also composed of several track segments coming from them. When there is not enough number of separate tracks, there will be problem of model construction. A solution to this is to simply break the tracks into several separate one with an editor.

5.4.4 Model construction

Figure 5.12 demonstrates what is done during the model construction phase. The part in black is the part that the original model has, and the part highlighted by color blue is the change in the model from this project.

The model construction is mainly done through class `CadModelBuilder` (the original structure please refer to section 4.2.7. A `ControlCenter` is first generated at the beginning phase of model construction as a child model of the `TopLevelModel`.

The model starts reading the information in the AutoCAD file from blocks called "source". When a "source" is found, which is in fact the exit of the depot, the model searches for the depot that it belongs to. If the name of the source ends with a number, then the name of the depot is the name of the source without the last number; otherwise they are the same. If the depot exists already, an extra *EndNode* will be added to the depot, a new child model "exit" will be created with its end node coupled with the end node of the depot. If the depot does not exist yet, a new *Depot* will be created, along with a *DepotControlUnit* as its child model. This new *Depot* is then externally coupled with the control center. Then the "exit" is added to it. Because now the exit is a child model of the depot, so its end node cannot be directly coupled with the track connected to it. In order to link the nodes, the EndNode of the depot, which is externally coupled with the EndNode of the exit, is linked to the start node of the next track. The coordinates of the depot is set according to the coordinates of the begin point of the first track that is connected to this depot. See figure B.13.

When a entrance is found in the file, the model searches for the depot that it belongs to, according to the same naming method as that of the `Exit` and does the similar thing

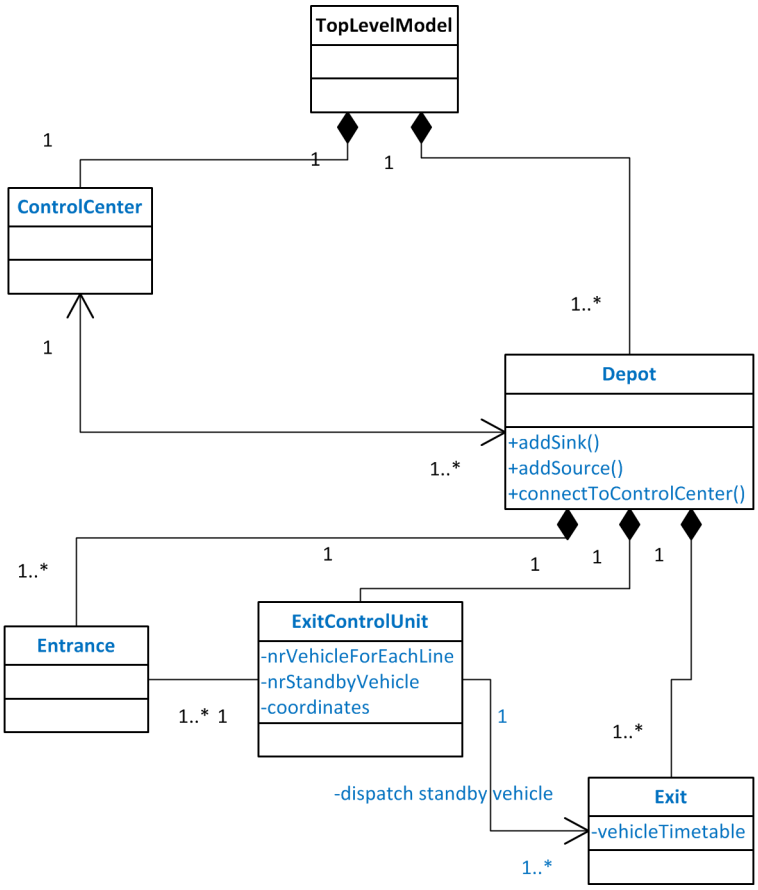


Figure 5.12: Model construction phase (depot all blue, source)

as it does for the exit. The *StartNode* of the depot is coupled with the StartNode of the entrance, and linked to the last track before the entrance. In this way, the exits and entrances with the same prefix of the name are added to the same parent model. See figure B.14.

Then the model keeps on searching for the next infrastructure connected one after another from the model tree (explain the model tree). When an entity does not belong to the *entityGroupMap*, it means that this entity is not a part of any stop, cross or a source. In this way, the parent model of the track will be directly the `TOP_LEVEL_MODEL`. For the sake of simplicity, we assume that disturbances can only happen around these kinds of tracks.

The pieces of track segments where disturbances can happen are marked by a different color in the AutoCAD file. They are read during model construction and stored in a list of disturbance entities. See figure B.15. Later when constructing the model tree, if the entity can be found from the *disturbanceTrackEntities*, then the track which is generated with the information through this entity can be used to generate disturbance. Since class `TrackSement` does not store the information of coordinates, a list is created to store the coordinates.

Only the track segment are put in the model tree so that they can be searched. As

a result, in order to find out which entrance is linked to the each of the line, a new method is used. When reading the AutoCAD file, all the entrances are added to a list. When coming to the end of the model tree, which means that no next element can be found anymore, the coordinates of the end of the last track will be used to search for the linked entrance. If the distance between a certain entrance and this end point is smaller to a "SNAPPING_TOLERANCE", it means that this entrance is linked to this end point in the AutoCAD file, then the information stored in this block will be used to generate the corresponding entrance in the model. See figure B.16

After an interlocking model is created, it needs to be configured to add the routs for different vehicles. The `DeadheadRouteReader` is used here to read the deadhead routing of the vehicles.

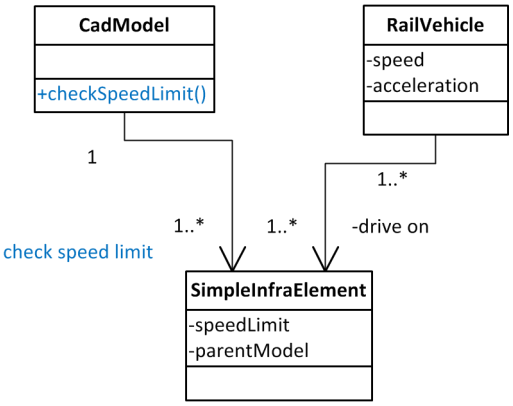


Figure 5.13: Model construction phase: check speed limit

Figure 5.13 shows that the speed limit of the infrastructure is then regulated again by method `checkSpeedLimit` in `CadModel`. It is regulated according to the whether it is in a stop, an intersection, and the speed limit of its next infrastructure. This is because, due to the structure of the model, the vehicles can only know the speed limit of the next infrastructure, despite of its length. Without this regulation, it could happen that if the difference between two track segments is too large, and the track segment linking these two is too short for the vehicle to decelerate to the lower speed limit, then the vehicle has to drive beyond the speed limit. However, in the original model, if the current infrastructure is followed by an intersection, the speed limit is regulated so that the vehicle can reduce its speed to zero until it reaches the next infrastructure. This is incorrect, because the speed limit should be regulated so that the vehicle can reduce its speed to the speed limit of the next infrastructre. And this problem has resulted in the overall lower running speed of the vehicles. This part of the model has been changed accordingly.

5.4.5 Initialization

After each `Exit` object is created, the *InitialValueReader* loads the information from the "initialValue" file to the exit. According to the information of the number of vehicles the exit has for each line and of number of stand-by vehicles, the `VehicleGenerators` with the timetable to dispatch vehicles are created and added to the exit. The InitialValueReader will also construct the vehicle schedules according to the sequence numbers

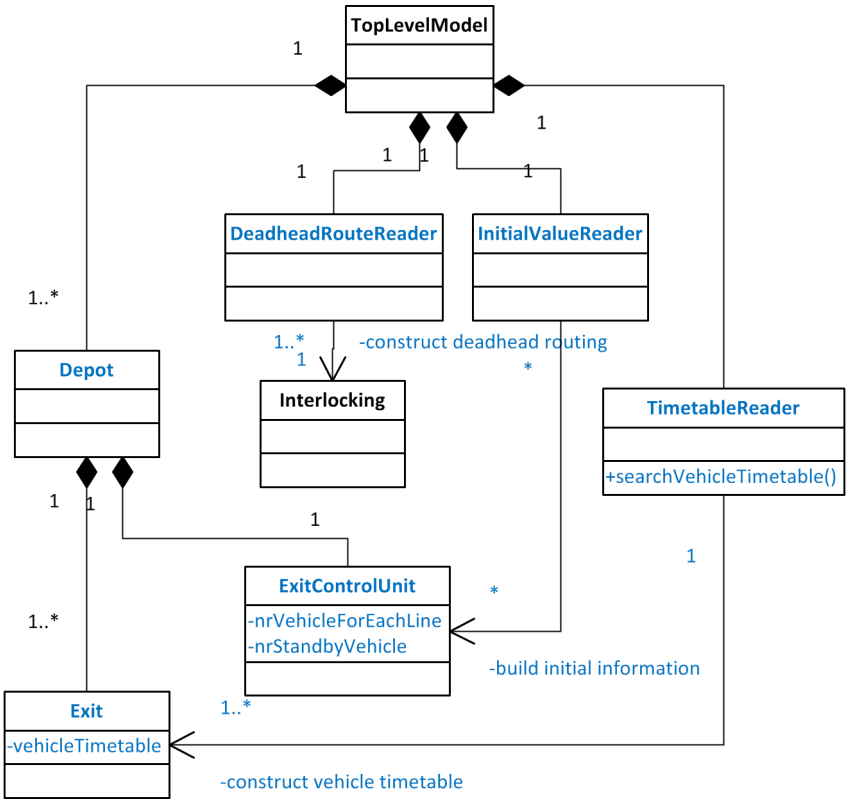


Figure 5.14: Initialization phase

specified in the "initialValue" file. These vehicle schedules with sequence number are stored in the exit. See figure B.20.

A new class *DeadheadRouteReader* is defined. Similarly as in the original model, the changes of the switches are defined by the route of the vehicle. However, here the route of the vehicle is not only defined by its line number, but also by its destination. This is because according to different depots the vehicles belong to, their deadhead trips can be different although serving the same service route. As a result, the two parameters, the line ID and the starting point, are combined into one *String* which is used to search for the routes. See figure B.19.

In `TimetableReader`, method `searchVehicleTimetable` is used to construct the vehicle schedule from the service timetable file. The objective is to construct a list of starting time of service trips for each vehicle of the line. When this is constructed, the line number is used, in order to search for the timetable of both directions. Searching in those rows, when a new cell with a numerical data type, it is the vehicle sequence number, which is used to distinguish the different vehicles of the same line starting from the same direction. If the timetable of this sequence number already exists, it will convert the time into the standard form and add this into the timetable and sort it, which will put it according to the time. It the timetable of this sequence number doesn't exit yet, it will create a new timetable and add this element. Schedules that are later than mid-night is ignored here, because if we add it directly and sort it, it will be added directly at the beginning of the timetable, the model will regard it as early in the

morning. If the start time of the model is 4 am, a timeadvance will be calculated as a negative number, which will not give the result as we want. See figure B.18.

5.4.6 Normal running

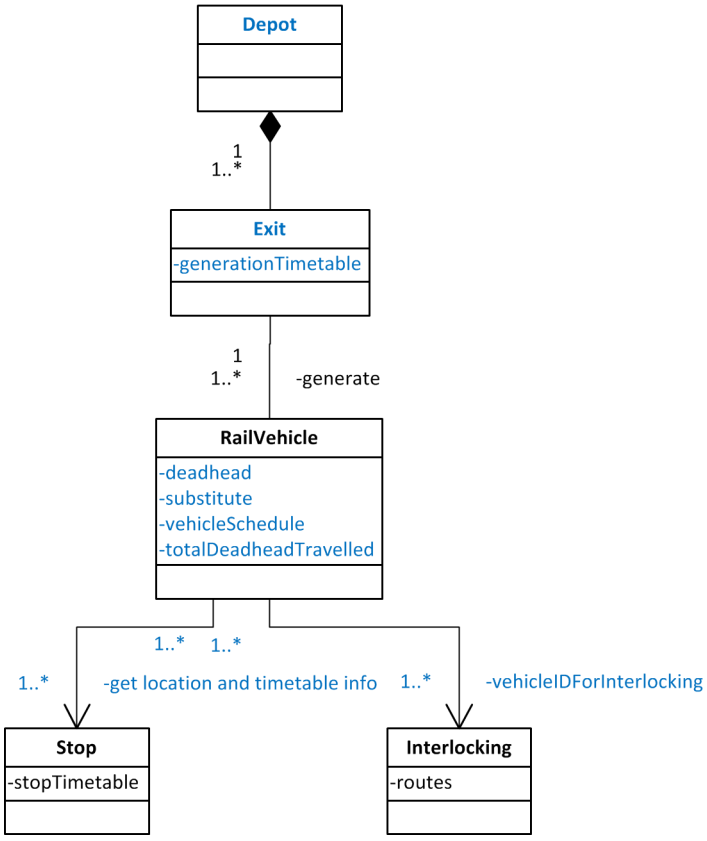


Figure 5.15: Normal running phase

Each `Exit` represents an exit of the depot. Vehicles are generated by `Vehicle-Generators` according to the timetable stored in them. When first generated, their attribute *deadhead* is set to true. They drive to the beginning of the line, wait at the stop till the time comes and start driving. This time is estimated approximately, since it does not affect the system significantly as long as vehicles do not halt at the stop for too long time and other vehicles have to queue after this one. Each `Entrance` represents an exit of the depot, where vehicles are deleted from the network after entering the depot.

When a vehicle is generated, the `VehicleGenerator` will assign it a sequence number, which includes both the direction number of the line and the vehicle's sequence in this direction. This number is used to obtain the vehicle schedule. Each vehicle schedule is constructed with a corresponding sequence number when the timetable is loaded into the exit.

It is difficult to specify the sequences that vehicles arrive at each stop. For example, for line 1 vehicle starting from Delft Tanthof, there are vehicles that start the trip after serving the trip after service for the direction from Scheveningen, and there are

also vehicles that come directly from another depot. It it difficult to predict the exact sequence that vehicles arrive at the end of Tanthof. If the vehicle which is supposed to serve the early trip arrive later than another vehicle, and the other vehicle still waits at the beginning of the line till the begining time, then the vehicle waiting after this will be late for all the stops. This problem is usually solved through communication and coordination between the drivers of the vehicles, but it is difficult to address in the simulation model. As a result, in order to simplify the problem and to reduce the distortion to the result, the standard service time of each vehicle is defined by the sequence that it reaches a certain stop. So in this case, even though the second vehicle was planned to serve the trip after the first one, since it arrives at the stop earlier than the other one, the stop will give its starting time according to the sequence that it arrives at the stop, which is the early one. So the vehicle will leave the stop when this time arrives, and the first vehicle will take on the service time which belonged to the second vehicle.

5.4.7 Disturbance

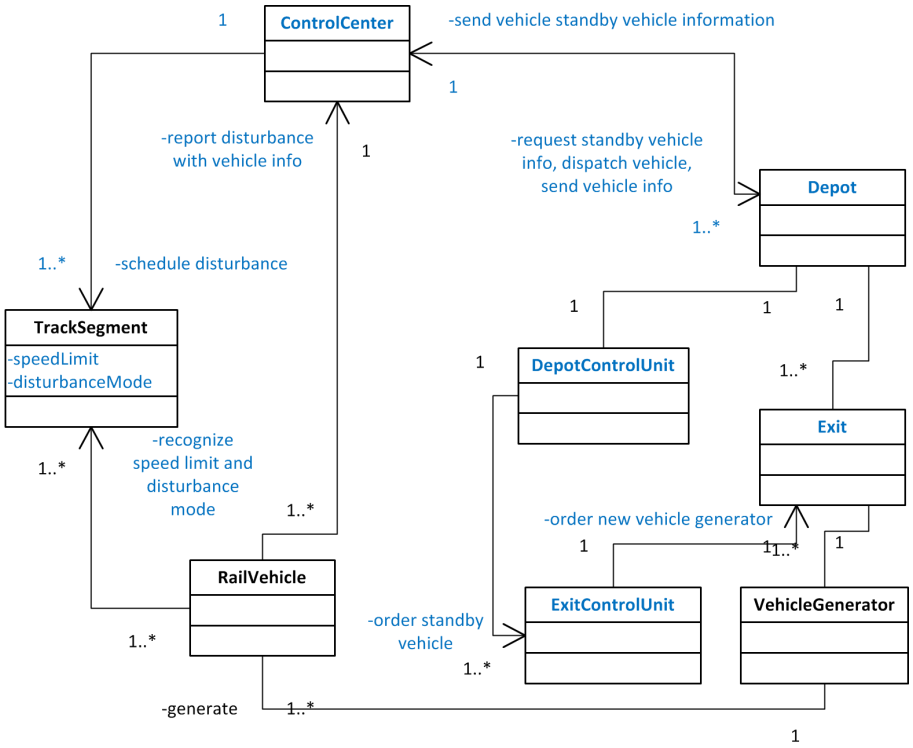


Figure 5.16: Disturbance phase

A random number is used to generate disturbance when vehicles breakdown. The probability of vehicle breakdown is calculated according to the time elapsed and possibility of breakdowns happen to a vehicle, which is calculated from historical data. When a problem occurs to the vehicle, the vehicle will first decide to stop, which is to change its phase to STOP.

In Control Center, every time a random number is set to decide if an accident is going to happen in the next hour or not. If there is going to be a disturbance, another

random nummer is used to decide at exact which time in the hour that the disturbance is going to happen. Another random number is used to select which area (near which track segment) will the disturbance happen. After these are all chosen, the control center will set the speed limit of the chose track to be zero. In this way, when the vehicle going towards the track, it will automatically stop in front of it. An attribute of the track "disturbanceMode" will also be set to true, which tells the vehicle that it will break down here and should delete itself from the model. A small slot of time advance is set to make sure that the control center collect information from all the depots without having to count the number of messages received. After receiving all the information, the control center will use a loop to go through all the messages to find out the nearest depot with available stand-by vehicle. The control center receives report of disturbances from the vehicles. After the period of time it takes to resume the service, the control center will send message to the vehicle behind the track segment which was waiting. Then vehicle can start to drive again. Much other information needs to be corrected before the vehicle can start driving again, including the information stores in the track segment about the vehicle which was deleted. The exit speed limit needs to be adjusted, because when the vehicle stopped, the speed limit is set to infinity, and this will cause problem in the computation of movement of the vehicle to start.

When the vehicle reports disturbance to the control center, it sends information including the location of disturbance and all the service information from the vehicle that the stand-by vehicle needs to serve the rest of the service of the vehicle. This is done through a class `VehicleInfo`, which can make sure that all the information needed is sent. The location is recorded as the last stop of the vehicle passed by, which is used to guide the vehicle to resume the service from there. After a period of time of disturbance to the network, the `VehicleGenerator` uses the information from the `VehicleInfo` to create a new 'identical' vehicle, whose attributes are defined according to the previous vehicle which was deleted and sets the *substitute* of the new vehicle to be true. In this way, the vehicle will request routes from the interlockings according to its *substituteLineBeginning*, and it change its state even if it comes across the stops that have the same name as its *lineBeginning* or *lineEnd*.

When a vehicle reaches a *TrackSegment* which has a speed limit of 0 and is in *DisturbanceMode*, it means that the vehicle will break down in front of the this part of track. The vehicle will send a message to the control center with all of its information, including service time table, line number, and other necessary information for the stand-by vehicle to keep on with its service. See figure B.12.

5.4.8 Data output

Delays of service will be calcuated according to the real time when vehicle arrive at certain stop and the standard time from the timetable.

Status of vehicles in the model are recorded into the database through out the entire simulation, including the cumulative total distance travelled and deadhead-kilometer travelled.

All the output data is recorded into a SQL database. It is necessary to perform data query to extract the necessary information from the database. For example, distance and deadheaddistance are two states of the vehicle that are recorded in the database through out the simulation and they are accumulative in calculation. If we want the final result of the distance, we need to get the maximum distance and deadhead distance from records with the same vehicle ID. Besides, in order to get the delay of service for every time a vehicle halts at a stop during a service trip, we should select the smallest

delay time (which is non-negative) for each stop and each tripID.

Calcualtion of extra dwell time at the beginning of the line: extra dwell time is the different between the arrival time (when the vehicle reaches the end of the stop) and the service time for this trip if the arrival time is before the service time. If the extra dwell time is long than the dwell time, it means that the vehicle will stay at the stop for longer than it is supposed to do for the passengers to get on, this is no need to add the normal dwell time to this part. However, since we want to separate the dwell time which is norally for the passengers to get on and off and the extra time that the vehicle spend at the beginning of the line, the amount of `dwellTime` is substracted from the `extraDwellTime` in the model, and the final waiting time of the stop is the sum of these two variables.

5.5 JAVA Code Implementation

The example below shows the code of class `ControlCenter`, which gives an example of the coding in JAVA with LIBROS-II library.

```
public ControlCenter(CoupledInfraModel parentModel) {
    super("ControlCenter", parentModel);
    // TODO Auto-generated constructor stub

    sigma = 0;
    this.initialize(0);
}
```

Figure 5.17: Example code: declaration of `ControlCenter`

Initialization of `ControlCenter` (figure 5.17):

The parent model is specified in the parenthesis. ``` ``ControlCenter'' ``` gives the `model name` of the model. `this.initialize(0)` specifies that the model will be initialzied at time 0, which means that the output function and internal transitions will be executed at time 0.

```java
@Override
protected void deltaExternal(double e, Object value) {
    // TODO Auto-generated method stub

    if(value instanceof MessageReportDisturbance){
        vehicleInfo = ((MessageReportDisturbance)value).vehicleInfo;
        reportedTrackSegment = ((MessageReportDisturbance)value).trackSegment;
        collectStandbyVehicleInfo = true;
        sigma = 0;
    }

    if(value instanceof MessageToControlCenter){
        msgList.add((MessageToControlCenter)value);
        timeReceiveMsgFromDepot = getTimeNow();
        sendContingencyPlan = true;
        this.sigma = 0.1;
    }
}
```

Figure 5.18: Example code: external transition functions of `ControlCenter`

External transition functions (figure 5.18):

If the incoming message is a message of `MessageReportDisturbance`,
obtain the vehicle information from the message,
specify the track of disturbance,
state change to `collectStandbyVehicleInfo`,
time advance is 0.

If the incoming message is a `MessageToControlCenter`,
add this message to the message list,
record time to receive the message,
state change to `sendContingencyPlan`,
time advance is 0.1.

```
protected void lambda ()
{
    if (disturbanceStart){
        disturbanceCounter++;

        TrackSegmentAndCoordinates tac = disturbanceTrackSegments.get(
                new Random().nextInt(disturbanceTrackSegments.size())
                );
        TrackSegment track = tac.track;
        track.originalSpeedLimit = track.getSpeedLimit();
        track.setSpeedLimit((float)0);
        track.disturbanceMode = true;

        disturbanceStart = false;
        this.sigma = Double.POSITIVE_INFINITY;
        return;
    }
    if(collectStandbyVehicleInfo){
        reportedTrackSegment.disturbanceMode = false;
        outputToDepot.send(new MessageToDepot(null, 0, null));
        collectStandbyVehicleInfo = false;
        sigma = Double.POSITIVE_INFINITY;
        return;
    }
```

Figure 5.19: Example code: output functions of `ControlCenter`

Output Functions (figure 5.19 and 5.20):

If is in state `disturbanceStart`,
disturbance counter plus 1,
choose one track randomly from the list of track segments,
save the original speed limit,
set speed limit as 0,
set the track segment in disturbance mode,
set the control center in passive mode.

If is in state `collectVehicleInfo`,
set track segment in normal mode,
send a message to depot with number of vehicle to dispatch as 0,
set the control center to passive mode.

```
if (sendContingencyPlan
        && DoubleCompare.greaterThan(getTimeNow(),
                timeReceiveMsgFromDepot)) {

    MessageToDepot msgToDepot = calculateContingencyPlan(reportedTrackSegment);
    msgToDepot.vehicleInfo = this.vehicleInfo;
    outputToDepot.send(msgToDepot);
    reportedTrackSegment.disturbanceMode = false;

    sendContingencyPlan = false;
    disturbanceFinish = true;
    this.sigma = (new Random().nextDouble() * 2 + 1) * 60;

    return;
}

if (disturbanceFinish) { // when the disturbance is finished

    if (reportedTrackSegment != null) {
        resetSpeedLimit(reportedTrackSegment);
        reportedTrackSegment.clearObjectsOnMe();

        this.outputToVehicle.send(new MessageReportDisturbance());
        reportedTrackSegment = null;
    } else
        System.out
                .println("Something wrong with reported disturbance track segment");

    disturbanceFinish = false;
    nextDisturbance = true;
    sigma = 60 * 60;

}
```

Figure 5.20: Example code: output functions of `ControlCenter`

If is in state `sendContingencyPlan` and time now is latter than the time of receiving message from depots

calculate contingency plan,
add information of the disturbance vehicle in the message,
send the message to the depots,
set time advance according to the distribution of length of disturbance,
set the state to `disturbanceFinish`.

If is in state `disturbanceFinish`,

reset the speed limit of the disturbance track segment,
clear the information stored in the track segment about the object that is on the track segment.
send message to the vehicle for the clearance of the disturbance,
set the state to `nextDisturbance`.
set time advance for next state.

```
@Override
protected void deltaInternal() {
    // TODO Auto-generated method stub

    if (nextDisturbance){
        if (disturbanceCounter < MaximumDisturbance){
            if (disturbanceHappen()){
                nextDisturbance = false;
                disturbanceStart = true;
                sigma = 0;
                return;
            }
            else //next time of checking if disturbance happens
                this.sigma = 60 * 60;
        }else //has reached maximum number of disturbances
            this.sigma = Double.POSITIVE_INFINITY;
    }

}
```

Figure 5.21: Example code: internal transition functions of `ControlCenter`

Internal transition functions (figure 5.21):

If is in state `nextDisturbance`,
    If total number of disturbances has not reached the maximum number,
    If the disturbance will happen in this hour,
        set state to `disturbanceStart`,
        set time advance as 0.
    Otherwise, set time advance as 1 hour to decide next time if disturbance will happen or not.
    Otherwise, set control center to passive mode.

6 Verification and Validation

Verification and validation is important to gain enough confidence on that the model can give a proper representation of what happens in reality. Verification is a quality control process, which checks if the specifications are correctly implemented by the system, and can be expressed by "Are you building it right?" While validation is a quality assurance process, which establishes evidence that a high degree of assurance that the product accomplished it intended requirements, and can be expressed by "Are you building the right thing?"

6.1 Verification

In this project, verification is mainly achieved through debugging. During this process the behaviors of the objects are checked step by step.

For example, the design of behavior of the vehicles at the stops is very difficult because of the complexity of the behaviors. Especially when they arrive at the first or last stop of the lines, their behaviors are very complicated and need to be carefully checked, which is discussed in section 5.3.3.

In this case, it is necessary to trace that when a normal vehicle comes to the first stop of the line, it changes *deadhead* into true, stops accumulating the distance of deadhead and starts to update the delay time. If a vehicle arrives early at the first stop, it should wait until the service time starts, and then record the delay time as "0". If the vehicle arrives late, it will still halt at the stop for a "halting time" and keeps on driving. The delay time here should be positive for this stop. When the vehicle arrives at the last stop of its trip, it still updates the delay time. Right after it leaves the last stop, the vehicle sets *deadhead* into false, starts to accumulate deadhead-kilometer, exchange the names of the first and the last stop of its service, and does not stop nor update delay time at stops until it arrives at the first stop of its next service trip.

This procedure does not only help the modeler to verify the behavior of the system, but also improves the modeler's understanding of the system. By following the behavior of the vehicles under different circumstances, the author realized that sometimes the behavior of the substitute vehicles are strange. Through tracing the behaviors of the vehicles step by step, it is found that vehicle starts halting at the stops after the first stop of the line instead of heading directly towards the stop where the disturbances happened without halting. As a result, an extra condition is added in the model so that the vehicle will only turns *deadhead* to true at the starting point of the line when its *substitute* is false.

A second method for verification is to check the data from the data base. This method can check if there is strange behavior of the system on a bigger picture. Because the model in this project is so big that it is impossible for the modeler to check all the behavior of the model step by step, the advantage of this method is that is can make sure that the system's behavior is logical on a macro level.

For example, through examining the delay times of a vehicles in a trip, it is found out delay time shows some spikes at certain stops in this trip. This is impossible because the difference between the delay times of consecutive stops cannot exceed the difference of the service times of those stops. This shows that there is a problem in the timetable from the stop. Through deeper inspection, the reason is round that in some timetable, the defined stop ID is wrongly taken as the stop from the opposite side with the same name. As stop ID is used for loading the service timetable information, the service time that the vehicles get that this stop is in fact not for this trip.

6.2 Walkthrough

In walkthrough, the author conducted an interview with a "Policy Advisor" of HTM. His main job content is about cost benefit analysis, optimization of transit system and simulation model of the system. As a result, he is familiar with the overall behavior of the system. He also understands the constraints from the perspective of research and modeling, therefore is able to give some constructive advices on both the model and the research topic.

In a walkthrough, a comprehensive review of what has done in the model is shown to the interviewee. In the process, the behaviors of the important aspects of the model are explained to the interviewee, including the behaviors of the `ControlCenter`, `Depot`, the interaction between `RailVehicle` and `Stop`. In addition, how these functionalities realized in the model is also explained to some extent. This process helps improve the understanding between the modeler and the interviewee. The assumptions of the model are also explained, and the interviewee have also suggested the possible consequences and solutions, which have been taken into modifications or limitations of the research.

The structure and the behavior of the control center are explained. The function of the control center mainly focuses on dealing with disturbances. The interviewee agreed on these aspects of the control center.

The behaviors of the vehicles in case of disturbance and at the ends of the lines are also explained. Originally when vehicles are at the first stop of the line, if the extra dwell time is larger than the amount of time that the vehicle should spent at the stop for passengers to get on, then the dwell time will be set to 0, then the sum of them will be the total amount of time that the vehicle spends at the stop. The interviewee proposed that the "dwell time" and "extra dwell time" should be clearly defined and separated in the model. To be more precise, dwell time should still be set to its original value, the its value should be deducted from *extraDwellTime*. This is because that, although for the modeler the final result will be the same, it is easier for the user of the model and the decision-maker to distinguish between the time that it is necessary for the vehicles to spend at the stop for the passengers to get on and off, the *dwellTime* in the model, and the extra time the vehicles spend between consecutive trips (*extraDwelTime*, which is in fact unnecessary for the system and can be reduced, eliminated or used for other purposes. This change is reflected in section 5.3.3.

The interviewee also raised the question concerning the behavior of the vehicles in case of disturbance. In the model, the substitute vehicle will go to the previous stop before the disturbance occurs and resume the service from there. However in real operation of the system, since normally it takes longer time for the substitute vehicle to come from the depot and then for the next vehicle in normal service to come to the stop, the substitute vehicles are often directly sent to the starting point of its next trip and resume the service from there. In some cases, if the substitute vehicle is delayed for too long, it may even be ordered to cancel the trip right after the one with disturbance, and to go to resume the service from the other end. However, it is very complex and time-consuming to simulate this behavior in the model, which does not have clear-cut rules to guide the decisions of the traffic controller, and it will make it very complicated for updating the delay time. But the interviewee also confirmed that this original design logical for the management of the system, and will not bring much difference. As a result, the original design is kept in the end.

The structure of the depot control unit and some modeling methods in `CadModelBuilder` were also explained to the interviewee. As a result of the walkthrough, we

can conclude that the behaviors and functionalities of the model correspond comply with the interviewee's requirements and expectations.

6.3 Compare the model results with the real data

In this section, I will compare the results given by the simulation model and the data collected from the real system. An analysis is given to show how the results given by the model and reflect the real situation, what are the possible reasons for the undesirable situations and what can be the possible solutions.

6.3.1 Validation of distance

Table 6.1 shows the trip distance of each line in the model and those from the real data. The trip distances calculated by the model differs for the two directions of the same line. The travel distances from different replications of the same service trip are exactly the same, as shown in figure D.1 for two directions of line 1. The real data

Line	From the model		From real data
	Direction 1	Direction 2	
1	20.114	20.065	20.1
6	11.256	11.144	10.7
9	13.556	13.516	13.5
17	16.200	16.820	16.5

Table 6.1: Trip distance from the model and real data (unit: km)

Table 6.2 shows the deadhead-kilometer (only the part of traveling between the depots and the ends of the lines under situation with no disturbance) and that from real data.

From depot	To stop	Result from the model	Real data
Scheveningen	Zwarte Pad	0.8	1
Scheveningen	Frankenslag	4.0	3
Lijsterbes	Abtswoudsepark	18.7	17.9
Lijsterbes	MCH Antoniushove	10.7	10.8
Zichtenburg	Leyenburg	2.1	2.4
Zichtenburg	De Dreef	3.9	4
Zichtenburg	Dorpskade	6.3	9

Table 6.2: Deadhead-kilometer from the model and real data (unit: km)

The difference between the deadhead-kilometer from the model and from real data is partly because of different routings. Reasons why the detail of routing of deadhead-kilometer are not defined exactly as it is reality. The deadhead trips are chosen by the author manually by looking for the a short route. This is also limited by the map of the network. Although all the tracked are available for service trips. But some other extra tracks are needed for the deadhead-trips, which are not yet in the network. However, we can still say that they are close enough.

The total distances from the model and from the real data are not compared because because the vehicles in the model do not follow the exact same vehicle schedules as in reality. However, the total distance traveled is composed of two elements: deadhead-kilometer and service trip distance. From both tables above, we can conclude that the model can give results that are close enough to reality on both elements, so we can have enough confidence on the distance calculated by the model.

6.3.2 Validation of time

Since delay time is an important performance indicator in this model, a way to gain confidence on the model is to compare the result related with time from the model and from real data. In this case, the average trip duration is a good indicator.

If average delay time is used, when there is a delay at the first stop and if the vehicle still drives with the normal speed, it is going to be delayed for the same amount of time for the rest of the stops in this trip, and the overall result will changed significantly; when the vehicle is delayed at the last stop, if there is enough buffer time before the next trip, then the vehicle is only going to be delayed for this one time, and the average result will change much less. As a result, although it should be totally random that when and where the disturbance happens in reality and in the model, final indicator could vary much. The measurement of trip duration will not be significantly affected by this type of factor. Besides, the allocation of number of vehicles to each direction also affect the result of average delay time. The comparison is shown in figure 6.1, where the x axis shows the number of vehicles for the two directions of line 1.

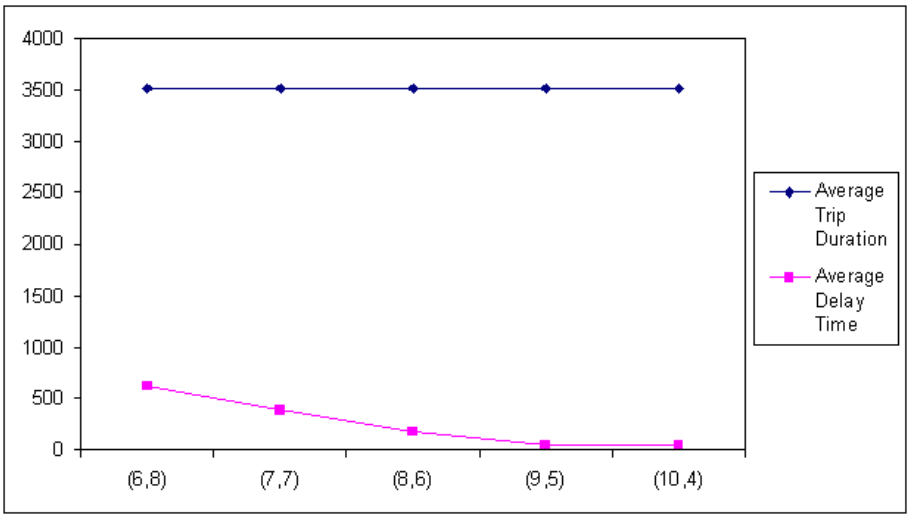


Figure 6.1: Change of average delay time and trip duration under different vehicle allocation of line 1 (unit: s)

As a result, although it is not a very interesting indicator of performance, the average trip duration is a good indicator to validate the result of the model. Table 6.3 shows the comparison of trip duration from the model and in reality.

The real data includes 40 weekdays selected between 7th September 2009 and 30th Octermber 2009. Results are each replicate are used to test if the confidence invertal includes the result from the real data. The defined trip duration is taken as the average

as the trip duration of the same direction of line varies during the day. The average trip delay of real operation is calculated from as the difference between the delay time of the first and the last stop. And the average trip duration from real operation is taken as the sum of this two elements.

	From the model		From the real data		Difference	
Direction	1	2	1	2	1	2
Line 1	3287	3523	3742	3275	7%	8%
Line 6	1951	1968	2097	1971	7%	0.1%
Line 9	2506	2465	2391	2328	5%	6%
Line 17	2872	3011	3163	3237	9%	7%

Table 6.3: Average trip duration (unit: s)

From table 6.3, for all the directions of the 4 lines in this study, all the differences between the results of the model and the results from real data fall below 10%.

A possible reason for this difference is that the driving behavior of the vehicles in the model is not the same as that in reality. In reality, The drivers can perceive the real-time situation, and decided to drive faster or slower in order to make the vehicles arrive on time. However in the model, the vehicles can only drive according to the specified speed limit, not faster or slower. As a result, we can expect that the average delay in reality is closer to 0 compare to the results from the model, and the variance of the real data should be smaller than that from the data of the model. The overall driving speed and the halting time could change among lines, due to difference characteristics of the lines, for example, the length, route, frequency, amount of travelers, etc. Due to time constraint, the author did not research further into this aspect, which is worth study in future research.

Statistical Test has also been conducted to test if the results from the model is significantly different from the mean of the real data. The detail in shown in Appendix D.2. Although the result shows that the result of the model is significantly different from the mean of the real data, according to the discussion above, we know that there is still limitation of simulating the driving behavior of the vehicles, and that the difference of all the directions are below a reasonable level, so we have enough confidence that the model can give a representative simulation of the real system.

7 Evaluation of the Impact of Capacity of Depots and Vehicle Planning on Key Performance Indicators

In this chapter, the analysis of impact of capacity of depots and vehicle planning on key performance indicators is presented. The detail of computer indicators is first given. The analysis of the decision of the capacity of the depots follows. The uncertainties involved in different aspects and the limitation of the study in this aspect are discussed. In this end, a method for a better vehicle allocation is suggested and the detail analysis is presented.

7.1 Computation of Key Performance Indicators

In the real data, the deadhead-kilometer is calculated only from the distance between the depots and the ends of the lines. However, according to definition, the distances that the vehicles travel between services should also be counted as deadhead-kilometer. Therefore, this part of trip is included in the model.

Two ways of computer the average delay time are used in the analysis. The first one is the average delay time, which is the numerical average of all the delay time. To calculate the second one, the negative delay times (when the vehicles arrive early) are taken as 0 and the numerical average is calculated from the data.

The first measurement is used because it shows the characteristics of the system without interference. However, if the objective is to reduce the average of delay time, it can also be achieved through an alternative with more negative values. It is undesirable that the vehicle arrive at the stops too early, and the usage of the first measurement may encourage this effect. With the second measurement of delay time, minimizing the indicator will not have the effect that we mentioned above. However, it also has the disadvantage that some part of the data is neglected. Because the monitoring of both measurements can give a more comprehensive representation of the data generated, both measurements are presented in the analysis in this chapter.

7.2 Comparison among Future Alternatives of the Capacity of the Depots

7.2.1 Input data

In order to make a decision for the future, the input data should be adapted to the future situation. Table 7.1 shows the capacity of each depot for the different alternatives. This information is provided by HTM. “Current situation” shows the capacity of the depots that HTM has now. This could be used as an comparison to show what will happen if HTM keeps the current situation. Alternative “With Remise Zuid” shows the alternative where depot Scheveningen and Lijsterbes will be closed, an emplacement Zwarte Pad, which is very close to Scheveningen, is used, and. “Without Remise Zuid” alternative shows the how would the company allocate the capacities of the depots if no new depot will be built.

Except from the capacities of the depots, many other inputs need to be obtained to simulate the situations in the future. Figure 7.1 shows the procedure of changing the input data for future analysis.

First, the service timetable is needed. It is the standard of the service and with which the delay time can be measured. However, currently no service timetable for the future is available from the company. The only thing we know about service timetable

	Current situation	With Remise Zuid	Without Remise zuid
Zwarte Pad/ Scheveningen	70	24	70
Remise Zuid		40	
Zichtenburg	52	58	58
Lijsterbes	42		

Table 7.1: Capacity for each depot for the alternatives

is the current version. As a result, the current service timetable is taken, but with some adjustments. For example, for line 1, currently all the vehicles are in fact dispatched from Scheveningen. However, it is a very long deadhead distance to drive from the end "Zwarte Pad", which is very close to Scheveningen, to the other end "Abtswoudsepark". Hence, HTM decided to turn this trip into a service trip. As a result, the service time from "Abtswoudsepark" is one hour later than that from "Zwarte Pad", because they are actually served by the same vehicle, which has to become from "Zwarte Pad" first before it can start the service from "Abtswoudsepark". However, if the future situations should be evaluated, the current timetable, which is especially designed for the current locations of the depots and vehicle allocations, is not proper for the evaluation. As a result, in the evaluation of future alternatives, the trips from both directions are assumed to start at about the same time, and the early trips are neglected, such as shown in figure E.1 and E.2 for line 1.

Second, in order to test different alternatives of the design of the depots, a common service timetable, or a service timetable that is design according to a standard that do not favor some particular designs, should be used. For the current situations, there are two sets of vehicle allocations available. The first one is what is happening in reality, for the adapted service timetable as mentioned above. The second one is for the timetable with no such adaptation (the ideal vehicle allocation). However, as the service timetable used for evaluating the future situations have been changed, it would not be fair to use the vehicle allocations designed for the real service timetable. And the change of service timetable is not done according some "ideal" rules of designing the timetable, it would also be unfair to use the "ideal" vehicle allocations. As a result, in order to create a common ground for the comparison between different alternatives, different vehicle allocations with the same number of total vehicles of the same line are tested, and the one with the least average non-negative delay time is chosen as the vehicle allocation to be under the alternatives, as shown in table 7.2. The directions of the lines are shown in figure E.1.

This is because that it is undesirable that the vehicles are too early or too late. If the least average delay time is chosen, it encourages option with large negative delay time, which is also undesirable for both the company and the travelers. Besides, the indicator of deadhead-kilometer is highly dependent on the locations of the depots that the vehicles are dispatched from, it is unfair to use the vehicle allocation, which is best for one design of the depots, to test another design.

Third, the optimization model is adjusted according to the new number of vehicles for each direction of the lines to get from which depot is each vehicle dispatched from. The result of different alternatives are shown in figure E.3, E.4, and E.5.

In the end, the time that the vehicles are dispatched from the depots should be

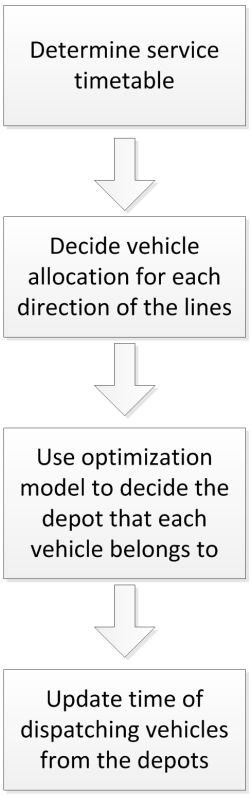


Figure 7.1: Procedure of changing input data for future analysis

Direction	1	2
1	9	5
6	8	1
9	12	3
17	12	3

Table 7.2: Number of vehicles for each direction of the lines

adjusted, according to the new distances between the depots and the starting points of the lines.

7.2.2 Results of the impact on the key performance indicators

The results of model for different alternatives are shown in table 7.3. The result of average delay and average non-negative delay is calculated as the weighted average according to the number of "stops" that vehicles make, which is the product of number of stops each trip and the number of trips per day.

Table 7.4 gives the results of indicator deadhead-kilometer of different alternatives. The results of the average delay time is not shown in terms of each line because they are sensitive to the location of disturbance and to which lines are included in the analysis. If the disturbance happen to line 1, then delay time will increase significantly of line

KPI	Current situation	With Remise Zuid	Without Remise Zuid
Deadhead-kilometer (km)	1044	1225	1454
Average delay (s)	56.8	46.3	57.1
Average non-negative delay (s)	92.4	91.2	92.7

Table 7.3: Results of model in year 2020

1. The increase is different depending on the when the location of disturbance is close to a certain depot or not. If there is a large number of different replications and it is possible to "generate" disturbance all over the network in the model, then this effect of uncertainty will not be big. However, since so far it is impossible and impractical to let disturbance happen in all the places in the network, we will only analyze the average delay will for the lines together.

Line	Current situations	With Remise Zuid	Without Remise Zuid
1	282	570	570
6	185	174	173
9	113	106	149
17	199	198	267
19	265	176	294
Total	1044	1225	1454

Table 7.4: Deadhead-kilometer per day of each line of alternatives given by the model (unit: km)

From table 7.3 and 7.4 we can see that, alternative "With Remise Zuid" is better than alternative "Without Remise Zuid' in every aspect. It is logical, because when added one more depot, the distance from one point to the closest depot will either stay the same or be improved. As a result, it can always improve on the unnecessary travel distance.

In comparison of the alternative of "Current situation" and "With Remise Zuid", each of them has three depots at choice. From table 7.4, we can see that except from line 1, all the other lines included in the study are improved on deadhead-kilomenter. The increase of deadhead-kilometer of line 1 is very obvious. As we have discussed before, that line 1 is special in the sense that it is its timetable is specially designed for the current situation. Even with the adjustment mentioned before, the service timetable from both directions are not the same, since then there are more trips starting from Abtswoudsepark. Line 19 benefits the most among the lines included in the analysis, because remise zuid is closer and easier to reach for both of its ends of the lines than other depots. However, many other lines with similar characteristics are not included in the study because they are operated with RR vehicles. Besides, from figure E.4 we can see that, there are also other lines that should benefit from the new depot, such line 12 and 15, but they are not yet included in the study. Statistical test has been conducted, which shows the results from the model of two alternatives are significantly different from each other (section E.2. In conclusion, it can be concluded that there is enough confidence that, on the two criteria, the means of the two alternatives are significantly different from each other.

KPI	Current situation	With Remise Zuid	Increase in percentage
Deadhead-kilometer (km)	1044	1225	15%
Average delay (s)	56.8	46.3	-18%

Table 7.5: Results of model in year 2020

As a conclusion, according to the analysis result so far with the five lines in the analysis, the alternative with Remise Zuid will benefit the performance of the network with a reduction of 18% of average delay time, but with the cost of increased 15% of deadhead-kilometer, as shown in table 7.5. The analysis is based on five lines in the network. The same model analysis method apply to the entire network, so future study with the input data covering all the operating lines can be conducted to obtain a more comprehensive analysis result of the transportation system.

7.2.3 Combined results with weights of the criteria

The SMART model (Simple Multi-Attribute Rating Technique) (section 3.3.2) is used here to assign weights to the criteria.

Because the different criteria is measured with different units, it is impossible to measure them together without any change. According to SMART technique step 7, the scores of the criteria should be normalized first. For each criteria, the lowest possible score among the alternatives is give score 0, and the highest one given 1. In the simplest way, the normalized score of the rest is calculated in proportion to the difference between it and the highest and the lowest scores.

Because criteria average delay time and average non-negative delay time (ANNDT) are dependent, only one of them should be used in this study. ANNDT is chosen here to illustrate how to use weights and to choose a robust alternative.

Ratings of the criteria are got only from the three "Policy Advisor"s from HTM due to lack of time and other constraints. In order to improve the weights given to the criteria, an actor analysis should be first conducted to find out the important people in deciding the alternative to be implemented. Their ratings on the criteria should be taken into consideration. In the meanwhile, there are also other actors can affect, indirectly, the choice of the decision makers, for example, the "Policy Advisor"s with their reports and suggestions. Other actors may also have the power to veto the final choice. What is more, it should be noticed that even the perception of the same person changes with time. So a decision maker may find some criteria more important than he thinks currently. As a result, the weights used in this section should only be taken as a example. An analysis on uncertainty of the weights is also shown later. A more detailed actor analysis and weights calculation should be conducted. The detail of rating of SMART technique is introduce in section 3.3.2. The ratings from the three "Policy Advisor"s are shown in table 7.6.

The final weights are taken as the average of the ratings. According to the multi-attribute utility theory (discussed in section 3.3.2, the values of the criteria are multiplied by their weights respectively, and the final score of each alternative is the sum of the products. Table 7.7 shows the results with the score given in table 7.6.

It can be concluded from thet able that, with the weights given by the policy advisors from HTM, the alternative "With Remise Zuid" outperforms the alternative with

KPI	Deadhead-kilometer	Average non-delay time
Policy Advisor 1	10	30
Policy Advisor 2	10	30
Policy Advisor 3	10	30

Table 7.6: Ratings on criteria from Policy Advisor of HTM

	Weight	Current situation	With Remise Zuid	Without Remise Zuid
Deadhead-kilometer	10	1	0.561	0
ANNDT	30	0.250	1	0
Total score		17.5	35.6	0

Table 7.7: Results of model in year 2020

the design of the current situations.

7.3 Comparison among the Alternatives under Uncertainty

7.3.1 Uncertain factors

No one can predict the future precisely. Yet people still want to go against the will of time, forecast what will happen in the future and make a decision now that could benefit us. If the an uncertain factor can highly change the future outcome, fewer people would take the risk; if the solution is robust enough against all the uncertainties, everyone will be happy to choose this one. Therefore, the first task is to see what are the uncertain factors, what are their possible values in the future and how much can they change the outcome of out decision.

The performance of a public transportation system is exposed to all kinds of external factors. One of them concerns the public: the income of the people determines whether they can afford a private car and how much they can spend on transportation, the preference of the people over different transportation mode (private car, bus, tram, train and flight). They are the customers of the transportation system and their choice determines the fate of the company. Another important external factor is political factor. HTM is financed by the trasit authority, which means that the company is directly affected by the political decisions of the goverment, such as development plan of the city, funding for public transportation development, obligatory performance standards. The change of technology can affect the price, speed and comfortability of the transportation modes, that could have an effect on the cost of the company and the preference of the travelers. And the economic factor, which has a profound impact on every aspect of the system, could exert a complex influence on public transportation system. Even within the company, the preference of the decision-makers, how they perceive the current situations the future possibilities also great affect the decision-making process. Due to the some characteristics of human beings (various interests, changing preference, bounded rationality), the human factor in the decision-making process is one of the uncertainties that we should not overlook.

All these external factors are so complex and entangled with each other that no

single model can solve the problem. In the scope of this project, some uncertain factors can be tested by varying the input parameters, for example, the design of the line, which should follow the development plan of the municipality, the frequency and duration of disturbances, which could be a result of changing technology of skills of the employees, and length of halting at the stops, which should be adapted to the demand of the travelers. However, since currently the element of the demand of the travelers is not yet included in the model, it is impossble to test the impact of the factors including the preference and financial situations of the customers, which could change due to both social and economical uncertainties.

The uncertainty of preferences of the decision-makers to different criteria prevails in all kinds of the decision-making processes. Due to time constraint, the result of the analysis is tested only under the uncertainty of the weights given to the criteria in the following analysis. In order to obtain a more comprehensive analysis of the results under uncertainties, factors mentioned in the previous paragraph (design of line, frequence and duration of disturbance, halting times) should also be analyzed with the similar method.

7.3.2 Analysis under uncertainties in weights of criteria

People always have their own opinions and interests. It is impossible to take the ratings of all the people. In addition, we have all experienced that even the opinion of the same person could change with time. Therefore, there is always uncertainty in the ratings collected in this way, and it is important to test how changes in the ratings could affect the final result. Table 7.8, 7.9 and 7.10 give the final score with different weights of the criteria, which are chosen arbitrarily.

	Weight	Current situation	With Remise Zuid	Without Remise Zuid
Deadhead-kilometer	10	1	0.561	0
ANNDT	10	0.250	1	0
Total score		12.5	15.6	0

Table 7.8: Score card under different weights 1

	Weight	Current situation	With Remise Zuid	Without Remise Zuid
Deadhead-kilometer	10	1	0.561	0
ANNDT	50	0.250	1	0
Total score		22.5	55.6	0

Table 7.9: Score card under different weights 2

From the results we can see that if we change the weights given to the different criteria. First, alternative with remise zuid is always the worst, because in fact it performs worse than both the other alternatives on both indicators. Second, if we increase the weight of deadhead-kilometer (or to decrease the weight of average delay time) to make the criteria with the same weights, the result still shows that the alternative with

	Weight	Current situation	With Remise Zuid	Without Remise Zuid
Deadhead-kilometer	20	1	0.561	0
ANNDT	10	0.250	1	0
Total score		22.5	21.2	0

Table 7.10: Score card under different weights 3

remise zuid is better the current situation. Third, if the weight of deadhead-kilometer is twice as high as that of average delay time, the design of current situation perform better than the alternative with remise zuid with a small margin. The performance of the alternatives under different weights is shown in figure 7.2.

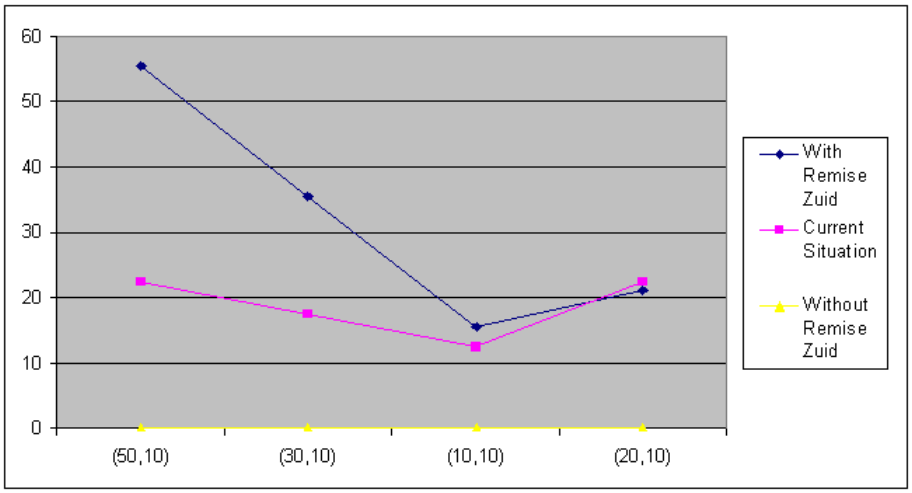


Figure 7.2: Performance of alternatives under uncertainty of weights

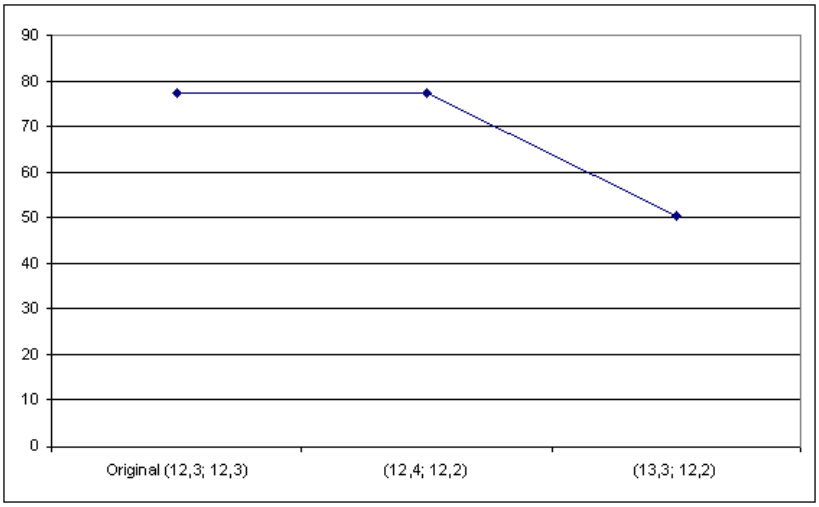
As a result, if we are confident enough that the average delay time is very important, which means that its weight can be around 5 (the highest weight can be given), then we have confident that the alternative with remise zuid can perform better than that in the current situations.

7.4 Vehicle Allocation

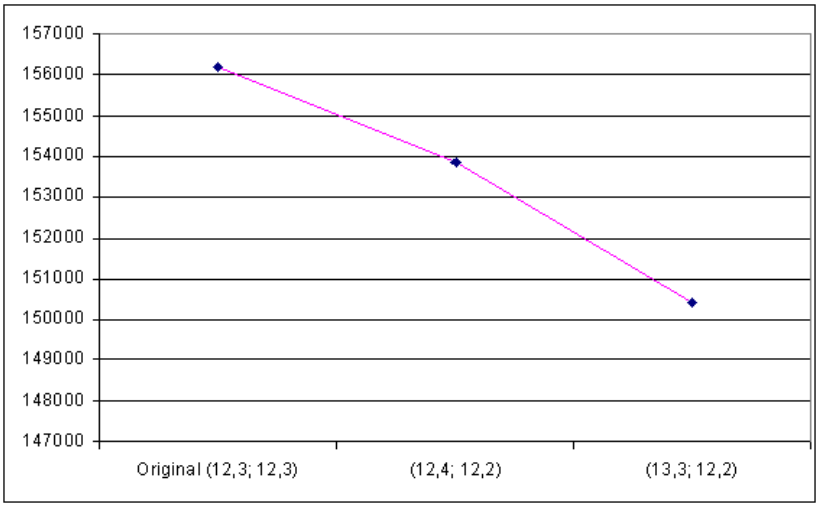
One important advantage of the simulation model is that it can used to study tactical decisions, such as vehicle planning. One important factor in vehicle planning is vehicle allocation. This includes the number of vehicles that are allocated to each direction of each line. As discussed before, vehicle allocation can significantly affect the performance of the system.

Take line 9 and 17 for example. Vehicle allocation discussed in section 7.2.1 is used as the base case. Since line 9 is relatively busier than line 16, the alternatives are to take one vehicle from line 17 and put it in line 9.

Figure 7.3 shows the performance of combinations of vehicle allocations of line 9 and 17.



(a) Deadhead-kilometer



(b) Average delay time

Figure 7.3: Comparison between different vehicle allocations: deadhead-kilometer (x axis: number of vehicles of line 9 direction 1, direction 2; number of vehicles of line 17 direction 1, direction 2)

	Nr. vehicle direction 1	Nr vehicle direction 2	Deadhead-kilometer (km)	Average elay time (s)
Line 9 (original)	12	3	113	141
Line 9	12	4	121	129
Line 9	13	3	114	77
Line 17 (original)	12	3	199	11
Line 17	12	2	187	23

Table 7.11: Parameters and separate result of the experiment

From the figure we can see that, the last vehicle allocation, which takes one vehicle of line 9 direction 2 and put it in line 17 direction 1, successfully reduces the total deadhead-kilometer and average delay time for these two lines. With this change, the average delay time is reduce by 30% and the deadhead-kilometer 3%. This important finding shows that balancing the number of vehicles allocated among the lines can improve the performance of the entire system without extra investment.

8 Findings and Conclusions

In this chapter, the research questions are firstly answered to give an idea to which extent this project solves the research problem. Then conclusions and recommendations to HTM are presented based on the findings from the research. Reflections about the research on the transportation system and on simulation modeling and suggestions for future study are given at the end.

8.1 Answers to Research Questions

1. What is the current situation of the transportation system, which kinds of impacts should be measured and what data is available for analysis?

   Currently HTM has five depots in the network. The company has decided to build a new depot by 2020, and some other depots will be closed. HTM has also purchased more vehicles, which will be in use in the coming years. Currently the depots are used for cleaning, maintenance and other operations to the vehicles and for storing the vehicles over night and during the days. The detail is given in section 2.1 and 2.2.

   HTM is interested in reducing the cost of the operations of the system and improving the satisfcation of the customers. Deadhead-kilometer and average delay of service are relevant indicators for these two aspect, and can be used to measure the impact of alternative designs of the depots. The current service timetable, future alternatives of the capacities of the depots and vehicle allocation according to the optimization model are available for the research. The detail is given in section 2.3.

2. What methods can be used to assess the performance of transportation systems and which one is the most appropriate in this study?

   Analytical, optimization and simulation models have been used to assess the impact of transportation systems. Simulation modeling is the most appropriate one for this study, because it can incorporate more detail of the system, for example the movements of the vehicles, the interaction among the vehicles and between the depots and vehicles and the exact service timetables. Simulation models can also give detailed representation of the behaviors of the system in case of disturbances. It is difficult for analytical and optimization models to include these types of detail. Simulation models can generate data on all kinds of aspects of the sytem, such as travel distance, instant speed, travel time. Simulation models can also be used for "what-if" analysis, which is very useful for the decision makert to improve their understanding of the system. This is answered in more detail in section 3.2.

3. How to design the tool to assess the impacts?

   The tool was designed with systems engineering methodology introduced in section 1.4. First, the requirements of the model are gathered and translated into the conceptual design with the relationships among some important classes, which is introduced in section 5.2. Then the important classes of the model are elaborated with more specified attributes and methods (section 5.3). In the end, the detailed design is conducted to realize all the required functions, which is presented in section 5.4.

4. How to validate the tool to ensure it reflects reality?

   In order to test if the tool reflects the reality, verification and validation need to be conducted. First, extensive debugging is used to trace the behavior of the model step by step on a micro level. By looking at the data stored in the database, it is possible to inspect the behavior of the model on a macro level. These two methods have been used to make sure that the model is built correctly (section 6.1). Second, a walkthrough is conducted with a policy advisor from HTM, who is familiar with the behavior of the transportation system and the expections of the model (section 6.2). Some assumptions were agreed on, and some details of ways of presenting the result were changed to help the decision-makers improve their understanding of the system. In the end, the results from the model are compared with the real data to gain enough confidence on the results given by the model, on both the direction of distance and time. The detail of the comparison and explanations of the differences between them is given in section 6.3. The conclusion is that the model is able to give an representative reflection of reality.

5. What are the impacts of the capacity of the depots and vehicle planning among the depots on deadhead-kilometer and average delay of service?

   Based on the analysis consisting five lines in the network, compared to the current design (Scheveningen, Zichtenburg and Lijsterbes), the future alternative with "Remise Zuid" depot and closing down "Lijsterbes" depot achieves 18% reduction on average delay time, but with a cost of 15% of increase in deadhead-kilometer. With the ratings obtained from the policy advisors of HTM, the alternative with "Remise Zuid" outperforms the design of current situation on the overall performance. Though the methodology applies to the entire network, the analysis results so far portraits the impact on these five lines. A more comprehensive study can be conducted with the same method and the input data covering all the operating lines. The detail is presented in secction 7.2.

   Balancing vehicle allocations among different lines can also improve the performance of the system without increasing cost. With the service timetable and vehicle allocation assumed in the research, a 30% reduction on average delay time and 3% on deadhead-kilometer have been achieved by allocating one vehicle from line 9 to line 17. The detail is given in section 7.4.

6. How will the system look like in the future, what uncertainty should we consider and how will it affect the analysis?

   Many factors involve uncertainty in the future, including social, political and economical factors. The model is able to test the effect of some of the factors by varying some input parameters, such as weights of criteria, frequency of service. Only uncertainty on weights of criteria was tested with the model due to time constraints. Some other factors, for example the development of the technology and the preference and the financial situations of the travelers, should have an impact on the performance of the system. However, it is difficult to reflect the impact of those factors because of lack of elements such as travelers' demand. Their impact and the method to select a robust alternative under uncertainties is suggested in section 7.3 and more suggestions are discussed in future study.

8.2 Conclusions and Recommendations

Strategic decisions have long term and large scale impact on the system, and constrain decisions on lower level. For example, the design of the capcity of the depots constrains vehicle planning and daily operations of the vehicle. Therefore, it is important to study the impact of the alternatives of strategic decisions before they are implemented. For the same reason, tactical decisions also constrain operational decisions. Hence good strategic tactical decisions can both improve the performance of transportation systems.

Deadhead-kilometer and average delay of service are relevant criteria. Deadhead-kilometer can reflect the cost of operating the system and average delay of service can directly influence the satisfaction of the travelers. Therefore they can be used to evaluate the performance of the system and can be used to assess the design of the capacity of depots and vehicle planning in the network.

Discrete-event simulation modeling is a suitable method for designing a tool to analyze the impact of capacity of depots and vehicle planning on deadhead-kilometer and average delay time. It models the detail behavior of the system, the interaction between components of the system, and provides a ground for a greater understanding of the transportation system.

Balancing the vehicle allocations among different directions of the lines can improve the performance of the system without new investment. Too many vehicles for one line can result in high deadhead-kilometer. Too few vehicles for one line will result in long delay time due to inability to recover from disturbance.

Spreading out the locations of the depots is beneficial. Currently some depots are too close to each other, and there is no depot available in the south part of the network. This situation results in long deadhead-kilometer and high delay time for some lines.

From the results given by the model, with key performance indicators of deadhead-kilometer and average delay time, if we are confident that the average delay time is very important, the alternative with a future remise zuid outperforms the design of the current situation.

It is difficult to assess different alternatives of design of depots with a service timetable that is specially designed for one specific situation. Fair and impartial argumentation requires a common ground for comparison among alternatives.

Some suggestions to improve the performance of the transportation system for HTM is listed below:

1. The internal work processes of the depots can be improved by a better training of the employees, more standardized work processes and more clarified job responsibility.

2. The numbers of vehicles assigned to directions of the lines should be balanced. The impact can be studied with the simulation model, and gain can be made without increasing cost.

3. It is good to spread out the locations of the depots. Adding the “Remise Zuid” depot can help the system achieve a better performance.

4. In order to have a fair analysis result of the impact, it is important to have “impartial” service timetable, which is not adapted to the condition of the locations of the depots.

5. The database should be enhanced towards the direction with a better integration. It can help improve the efficiency of the work.

8.3 Reflections

The entire project lasted more than 6 months. During this period, there have been mistakes, struggling, learning and improving. Below are the reflections from the author, which hopefully could be of some help to other people who are working on similar projects.

1. Try to find out what is the real problem and why people care about it. In the beginning of the project, I had a false opinion about what is the problem and was trying to develope a method to solve the problem which does not exist. It was until when I started going to the depot frequently, talking to different people from the company and from the depot that I realized that the problem lies somewhere else. Field study is important, and that is the way to find out the problem, if you want it to be realistic.

2. Transportation systems are way too complicated, especially when you are dealing with a real one. I like to learn from practical things, to see how people really work, instead of how things are written on the papers. It can always give you surprises, with new findings and more insights. What I do is to get out of the room and walk into reality. Surprises lie in real life.

3. In Chinese we say, "from any three people, there is alway someone who could be my teacher". Talking to people is a great way to learn and to broaden one's perspective. And communication is an important to improve the understanding. Always communicate in advance, warn the consequences and work together towards a better solution.

4. Planning in advance is always good. The planning can never be too detail. Planning has always been my weakness. Maybe some people have to feel the time pressure to work hard, maybe in the end you realize that it is never possible to perfect your work. But trying to foresee as much as possible allows more time to think and more time to reflect.

5. Programming libraries should be properly documented. There is no documentation on LIBROS-II model. It was time consuming to learn the concept and functionality of LIBROS-II model.

6. A standardized input can help improve the efficiency of work. There are many files used to prepare the input data of the simulation model. The output of one file is manually adjusted and used as the input of another file. I was too lazy to work on combining the files to generate the input in an automatic way. But now I have realized that it could have saved me much time and mistakes.

7. For big models, it is desirable to validate one part before adding another one. Otherwise, extensive debugging can be the only solution. Much time was consumed when it is difficult detect where does the problem come from.

8. Writing is never easy for me. It seems always difficult to make other people understand my logic in paper work. But I didn't stop trying. I am even surprised by my own improvement in writing. Language and ways of thinking can be learned in daily life. Pay attention to what happenes around, in class, in movie, in talking to people with wisdom.

8.4 Future Study

The model in this project is still limited by many factors. As a result, the following are suggested for further research on developing the simulation model and on assessing the different alternatives for future developments of the depots.

- The evaluation of future situations is actually based on the timetable of current situation. The dilemma it brings is that this service timetable is somehow "customized" according to the current situation, including the location of the depots. In order to provide a common ground for comparison among the alternatives, an "ideal" service timetable should be used, which is designed without the considerations of locations of the depots.

- Some depots have maintenance function and some do not. This difference has not been included in the study. All the vehicles should go to the maintenance depots after a period of time for preventive maintenance and other routine operations. However, the depots do not go back to the depot during the day in this model yet. A future analysis on this aspect can provide an insight of how the maintenance function influence the schedule of the vehicles, and thus on deadhead-kilometer and delay of service.

- All the routes in the model are currently defined manually be specifying the directions of all the switches all the way. This is a very inefficient way and also prevents the possibility of having "disturbances" at any possible place in the network. A future study on a better routing methods can improve the analysis of the impact of disturbances on the network.

- Drivers' behavior. In the model, the speed of the vehicles are controlled by speed limit. This means that, with give acceleration and deceleration, the vehicles will move as fast as they can. However, in reality, the speed of the vehicles are much more complicated than this. speed of the vehicles. Without an accurate representation of the actual behavior of the vehicles, the model cannot simulate, or is used to predict, the behavior of the system in reality. Detail of discussion on this topic is presented in section 6.3.2.

- The weights assigned to criteria are calculated from the ratings of the policay advisors from HTM to have a general idea about how people in HTM value those criteria. Besides the transportation company HTM, the transit authority, the municipality and the public all have impact on the system. A more comprehensive actor analysis and the resulted improved weights to the criteria can give a better indication of what really matter to different actors, and how can the system be improved for most of them.

- A public transportation system is exposed to many external factors, which are highly uncertain in the future, such as social, economical, technological and poticial factors. Uncertainties of some factors can be tested in the model in future analysis, such as frequency of disturbance, service timetable design and line designs. In order to the impact of other uncertain factors on the performance of the system, elements such as demand of travelers should be included. This is discussed in more detail in section 7.3.1.

- The current analysis is based on five operating lines in the network. The same model and research methods can be applied to the entire network. In order to gain

a more comprehensive of the impact, future study can be conducted with input data covering other lines, including the service timetable and vehicle allocation.

- Several input files are used in this project. Some of the have overlapping functions and can be in fact integrated. A future work on integrating the input data files can improve the efficiency of work and reduce the chance of some problems caused by human mistakes.

References

[1] R. Shishko and R. Aster. NASA systems engineering handbook. *NASA Special Publication*, 6105, 1995.

[2] H. Chestnut. *Systems engineering tools*. Wiley, 1965.

[3] A.P. Sage. Systems engineering education. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 30(2):164–174, 2002.

[4] E. Cascetta. *Transportation systems analysis: models and applications*. Springer Verlag, 2009.

[5] G. Schmidt and W.E. Wilhelm. Strategic, tactical and operational decisions in multi-national logistics networks: a review and discussion of modelling issues. *International Journal of Production Research*, 38(7):1501–1524, 2000.

[6] D.N. van Oort and O.C. Ware. *Service Reliability and Urban Public Transport Design*. PhD thesis, Delft University of Technology, 2011.

[7] E.R. Petersen and A.J. Taylor. A structured model for rail line simulation and optimization. *Transportation Science*, 16(2):192, 1982.

[8] AC Williams. A stochastic transportation problem. *Operations Research*, 11(5):759–770, 1963.

[9] A.A. Assad. Models for rail transportation. *Transportation Research Part A: General*, 14(3):205–220, 1980.

[10] D. Huisman, R. Freling, and A.P.M. Wagelmans. Multiple-depot integrated vehicle and crew scheduling. *Transportation Science*, 39(4):491–502, 2005.

[11] L. Yves et al. An exact algorithm for minimizing routing and operating costs in depot location. *European Journal of Operational Research*, 6(2):224–226, 1981.

[12] B.A. Murtagh and S.R. Niwattisyawong. An efficient method for the multi-depot location-allocation problem. *The Journal of the Operational Research Society*, 33(7):629–634, 1982.

[13] Tai-Hsi Wu, Chinyao Low, and Jiunn-Wei Bai. Heuristic solutions to multi-depot location-routing problems. *Computers & Operations Research*, 29(10):1393 – 1415, 2002.

[14] C. Barnhart and G. Laporte, editors. *Handbooks in operational research and management science*, volume 14. North-Holland, 2007.

[15] R.E. Shannon. *Systems simulation: the art and science*. Prentice-Hall Englewood Cliffs, New Jersey, 1975.

[16] D. Brunner, G. Cross, C. McGhee, J. Levis, and D. Whitney. Toward increased use of simulation in transportation. In *Simulation Conference Proceedings, 1998. Winter*, volume 2, pages 1169–1175. IEEE, 2002.

[17] A. Greasley. Using simulation to assess the service reliability of a train maintenance depot. *Quality and Reliability Engineering International*, 16(3):221–228, 2000.

[18] N. Adamko and V. Klima. Optimisation of railway terminal design and operations using villon generic simulation model. *Transport*, 23(4):335–340, 2008.

[19] L. Cheng and M.A. Duran. Logistics for world-wide crude oil transportation using discrete event simulation and optimal control. *Computers & chemical engineering*, 28(6-7):897–911, 2004.

[20] J.S. Hooghiemstra and M.J.G. Tunisse. The use of simulation in the planning of the dutch railway services. In *Proceedings of the 30th conference on Winter simulation*, pages 1139–1146. IEEE Computer Society Press, 1998.

[21] A. Nash and D. Huerlimann. Railroad simulation using OpenTrack. *Computers in Railways IX*, pages 45–54, 2004.

[22] M. Jian-rui, B. Yun, and Z. Jin-zi. An event-driven simulation model for the railway station. In *Computer Design and Applications (ICCDA), 2010 International Conference on*, volume 5, pages V5–189. IEEE, 2010.

[23] M. Ebskamp. Exploratory modeling: a promising method for flood risk management? Master's thesis, Delft University of Technology, 2009.

[24] S. Dessai and J.P. Van der Sluijs. Uncertainty and climate change adaptation: a scoping study. *Report NWS-E*, 2007-198(2007):1–95, 2007.

[25] C. Zegras, J. Sussman, and C. Conklin. Scenario planning for strategic regional transportation planning. *Journal of urban planning and development*, 130:2, 2004.

[26] M. Wang and J. Yang. A multi-criterion experimental comparison of three multiattribute weight measurement methods. *Journal of Multi-Criteria Decision Analysis*, 7(6):340–350, 1998.

[27] W. Edwards. How to use multiattribute utility measurement for social decisionmaking. *Systems, Man and Cybernetics, IEEE Transactions on*, 7(5):326–340, 1977.

[28] W. Edwards and F.H. Barron. Smarts and smarter: Improved simple methods for multiattribute utility measurement. *Organizational Behavior and Human Decision Processes*, 60(3):306–325, 1994.

[29] M. Poyhonen and R.P. Hamalainen. On the convergence of multiattribute weighting methods. *European Journal of Operational Research*, 129(3):569–585, 2001.

[30] V. Belton. A comparison of the analytic hierarchy process and a simple multiattribute value function. *European Journal of Operational Research*, 26(1):7–21, 1986.

[31] F.A. Lootsma. Scale sensitivity in the multiplicative ahp and smart. *Journal of Multi-Criteria Decision Analysis*, 2(2):87–110, 1993.

[32] J. Mustajoki, R.P. Hämäläinen, and A. Salo. *Decision Sciences*, 36(2):317–339, 2005.

[33] H.A. Simon. Theories of bounded rationality. *Decision and organization*, 1:161–176, 1972.

[34] P.H.M. Jacobs, N.A. Lang, and A. Verbraeck. Web-based simulation 1: D-sol; a distributed java based discrete event simulation architecture. In *Proceedings of the 34th conference on Winter simulation: exploring new frontiers*, pages 793–800. Winter Simulation Conference, 2002.

[35] P.H.M. Jacobs. The dsol simulation suite. enabling multi-formalism modeling in a distributed context. *Delft University of Technology, Delft, The Netherlands, Doctoral dissertation*, 2005.

[36] M.D Seck and A. Verbraeck. Devs in dsol: Adding devs operational semantics to a generic event-scheduling simulation environment. *Proceedings of the Summer Simulation Multiconference (SummerSim'09)*, 2009.

[37] B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, editors. *LIBROS-II: Railway Modeling with DEVS*. Delft University of Technology, 2010.

[38] A.I. Concepcion and B.P. Zeigler. Devs formalism: A framework for hierarchical model development. *Software Engineering, IEEE Transactions on*, 14(2):228–241, 1988.

[39] B.P. Zeigler, H. Praehofer, and T.G. Kim. *Theory of modeling and simulation*, volume 100. Academic press, 2000.

[40] B.P. Zeigler. Hierarchical, modular discrete-event modelling in an object-oriented environment. *Simulation*, 49(5):219, 1987.

[41] P.T. Eugster, P.A. Felber, R. Guerraoui, and A.M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131, 2003.

A Detail Description of Depot Zichtenburg

Currently there are 5 depots in the network of HTM. 3 of them are maintenance depots, where vehicle can be maintained and repaired because they are equipped with the necessary mechanics and equipment. HTM is planning to build another depot in the south part of the network, which is called “Remise Zuid” so far. The locations of these depots are shown in figure 2.1 with red dots. The rest are called emplacement, where vehicles can be parked over night and only small and simply operations to the vehicles are possible.

Right now there are several locations that HTM can choose from for the new depot. Although the company has a target capacity of the new depot, because of the limit of some location, the combination of more than one depot with smaller capacities is also optional. Another factor of the study is that whether the depot is used only for parking or also for maintenance area. This issue arises because that it is costly to equip the depot with maintenance equipments and staff, while centralizing those resources is less costly. However, then all the vehicles have to go to the same depot, no matter how far are the ends of the service lines are they from. And this way, it will also increase the complexity of the management of a big depot. If the system of one big depot shuts down, many more vehicles are affected than if they are distributed in several small depots.

Depot Zichtenburg is the biggest depot in HTM network. It is responsible for the maintenance work for the new RR vehicles. In Zichtenburg, there is a software monitor and control system, which is used to have an overview of the location of all the vehicles inside the depot and to change switches when the vehicles need to move.

The depot is divided into three parts. The first one is to place the GTL trams, as shown in figure 2.4. Only the first track in this part is equipped with a washing and a sanding machine, which is used for all the washing and part of the sanding operations of the vehicles in this depot. The second part of the depot is used to place the new RandstadRail vehicles.

Figure A.1: First part of depot Zichtenburg

The last part is used for major maintenance. In order to carry out a major maintenance task that needs special tools, the maintenance mechanics have to use a special car to drag the vehicle into the maintenance area. This is because, due to safety reason,

there is no power supply for the vehicles inside the maintenance area. For small maintenance, the vehicles can be left in the parking area and the maintenance mechanics can perform their job there.

All kinds of operations inside the depots are scheduled according to the vehicles planning for the exploitation services. A priority rule is that the operations within the depots should not interfere with the service exploitations. Since mostly operations (including sanding, washing and small maintenance) within the depots can be finished within a relatively shorter period of time compare to the time length the vehicles park inside the depots every day, and many vehicles can stay in the depot for a long time during off-peak hours (from 10 am to 3 pm and after 9 pm), this rule is strictly followed by postponing some operations if they cannot be finished surely before the exploitation.

In the future, Lijsterbes will closed down by year 2020, and Zichtenburg will remain its current capacity until 2016. Emplacements Leidschendam and Oosterheem will remain unchanged. The company decided to build 'emplacement Zuid' by year 2020, which will be chosen from one of the locations in the south part of the network: Opstelterrein Zegwaard or Opstelterrein.

Since all the new coming vehicles are the more advanced RR vehicles, which are wider than the GTL vehicles. So all the new depots are built according to the standard of RR vehicles, while the GTL vehicles can still use those standing places. In order to satisfy the need for RR vehicles standing places, some old tracks for GTL vehicles might be rebuilt to suit RR vehicles, resulting smaller capacity from the same area.

One of the possibilities is also to build a new emplacement at Zwarte Pad, which is near Scheveningen and can be expanded from an existing turning point. Because this place is very close to Scheveningen, which is a popular area for both tourism and business, the depot Scheveningen depot might have to be shut down. In this case, the only left maintenance depot Zichetenburg will be enlarged.

Another possibility, in alternative 3, is that no expansion at Zwarte Pad will be made. In this case, in order to satisfy the need of capacity for RR vehicles, the maintenance depot will be rebuilt so that 12 RR capacity is made from the area of 20 GTL vehicles. In this case, Zichtenburg depot will remain the same capacity as it has in 2016.

Above is the central control system, which is used to have an overview of the location of all the vehicles inside the depot and to change switches when the vehicles need to move. The depot is divided into three parts. The first one is to place the old type of trams, such as in the picture below. Only the first track in this part is equipped with a washing and a sanding machine, which is used for all the washing and part of the sanding operations of the vehicles in this depot.

The second part of the depot is used to place the new RandstadRail trams as in the picture below.

The last part is used for major maintenance. If for a major maintenance which needs some special tool, the stuff have to drag the vehicle by a special car into this maintenance area. This is because due to safety reason, there is no power supply for the vehicles inside the maintenance area. For small maintenance, the vehicles can be left in the parking area and the maintenance stuff perform their job over there.

Figure 1 Special car to drag the vehicles All kinds of operations inside the depot are scheduled according to the vehicles planning for the exploitation services. A very important rule is that the operations inside the depot should not interfere with the service exploitation. Since mostly operations inside the depots can be finished within a short period of time (including sanding, washing and small maintenance), and many vehicles can stay in the depot for a long time during off-peak hours (from 10 am to

3 pm and after 9 pm), most operations inside the depot do not interfere with the exploitation process. However, a problem interfering the service process is related with maintenance stuff. During the interview, two similar accidents happened. In the first time, in order to take the vehicle out of the maintenance area, the special car is needed to pull out the vehicle. However, the sensor on the tracks cannot recognize the special car, thus for safety reason, the whole track system was shut down. At this time, the controller should examine the situation, make sure nothing dangerous is going on and then reset the system. However, before the system could be reset, if some vehicle wants to go out for service, the switches cannot change automatically because of the shutting down of the system. It will take up to 10 min for the driver to change 4 or 5 switches so that the vehicle can get out of the depot. This will result in delay for service for this vehicle. In the accident, one maintenance stuff drove the vehicle backwards out of the maintenance area without notifying the central controller, causing the sensors to give warning and shut down the system again. And another vehicle was delayed again for the same reason.

There is another problem for the management of the depot. As the HTM are getting more vehicles, the depot Zichtenburg is required to handle more vehicles than before. The maintenance area is becoming inadequate and sometimes vehicles are also parked outside the garage.

A.1 Detail of the Plan Function

On the strategic level, a decision has been made that a new depot is going to be built. However, the capacity of the new depot, which of the current depots should be closed or kept are still not clear. On this level, different alternatives are assessed according the evaluation of different criteria that the company is interested in, such as deadhead-kilometer, robust of service, etc. In fact, tactical and operational decisions, which are on lower levels than strategic decisions, will also affect the result on those performance criteria, and they need to be decided with decisions on strategic level. For example, the vehicle schedule can only be made with the information of capacities of all the depots. As a result, in order to evaluate the strategic decision, each alternative should be evaluated according to the optimized tactical and operational decision that will be made under the strategic solution.

The current planning process in HTM can be divided into three steps: timetable planning, vehicle planning, and personnel planning. These three steps of planning all affect the efficiency of the system and are interrelated with each other. Timetable planning includes the planning of timetables for travelers, which are the ones the travelers use when they plan their traveling and thus have direct impact on travelers. Travelers usually prefer to have a stable timetable so that there is no need to plan the trip everyday. As a result, the timetable usually stays the same for a long period of time.

Next, the vehicle planning is made according to the timetable from the previous step. It includes the scheduling of vehicles, that is, which exploitation process is the vehicle scheduled for, and when and to which depot should the vehicle go for maintenance and parking. Vehicle planning should support well the timetable planning because timetable is relatively stable and faces the travelers directly. It is also important to assess the vehicle planning because some criteria of the transportation system's performance can be concluded to support the decision-making process. For example, deadhead-kilometer should be reduced as it measures the distance that vehicles have to travel without providing service. It is also desirable to reduce the delays that travelers experience in the transportation system.

In the last step, personnel planning is designed to support the vehicle planning. It is to plan the schedule of the staff required for all kinds of operations according to vehicle planning, including driving of vehicles, maintenance inside depot and so on. This step is mainly affected by vehicle planning, but also can also cause influence the other round. For example, if it is impossible to schedule the staff according to the vehicle planning, it has to be changed until a feasible solution is found.

HTM is now intending to build a new depot in the system with different alternatives of capacity to improve the management of vehicles. The vehicle planning will be changed when new depots are in use for parking, maintenance and other operations. The location of depots directly affects deadhead-kilometer and the capacity limits how many vehicles can be scheduled in the depot at the same time. As vehicles are dispatched from depots, the location and capacity of depots affect how vehicles can be rescheduled in case of delays of operations, thus affect the delays of services provided to the travelers. As a result, in order to select a good solution of the location and capacity of the new depot, we should examine the deadhead-kilometer and delay of service of the vehicle plannings under different scenarios of each alternative.

A.2 Accidents

In the first time, a mechanic was trying to take a vehicle out of the maintenance area. For safety reason, there is no power line for tram vehicles inside the maintenance area, so this job has to be done by a special car. However, due to some compatibility problem, the sensor on the tracks cannot recognize the special car. For the software system of the depot, this situation is recognized as “abnormal”, thus for safety reason, the whole track system was shut down. At this time, the controller at the central office needs to examine the situation, make sure nothing dangerous is going on and then reset the system. However, before the system could be reset, if some vehicle wants to go out for service, the switches cannot change automatically. It will take up to 10 min for the driver to change 4 or 5 switches so that the vehicle can get out of the depot. This will result in delay of service for this vehicle. In the second accident, one maintenance mechanic drove the vehicle backwards out of the maintenance area without notifying the central controller, causing the sensors to give warning and shut down the track system again. Another vehicle was delayed again for the same reason.

B Documentation of LIBROS-II and New Development

B.1 Atomic Model

The output of the atomic model is send by using method `outputPort.send()`. When an external event occurs, `activePort()` gives the port of the model which receives the incoming message, the `value` in the `deltaExternal` method gives the value of the incoming message.

B.2 InfraComponenet

`InfraElement` is a kind of `AtomicModel`. It has the most basic functions as an infrastructure element. It has a speed limit. A `InfraComponent` is a kind of `CoupledModel`. They can all have several start nodes and ends nodes. A `Simple-InfraElement` and a `SimpleInfraComponenet` are the objects with only one start node and one end node.

An `InfraComponent` is a coupled model, which has a speed limit, start nodes and end nodes. A `SimpleInfraComponent` is an `InfraComponent`, which has only one start node and one end node. The relationship between these classes is shown in figure B.1.

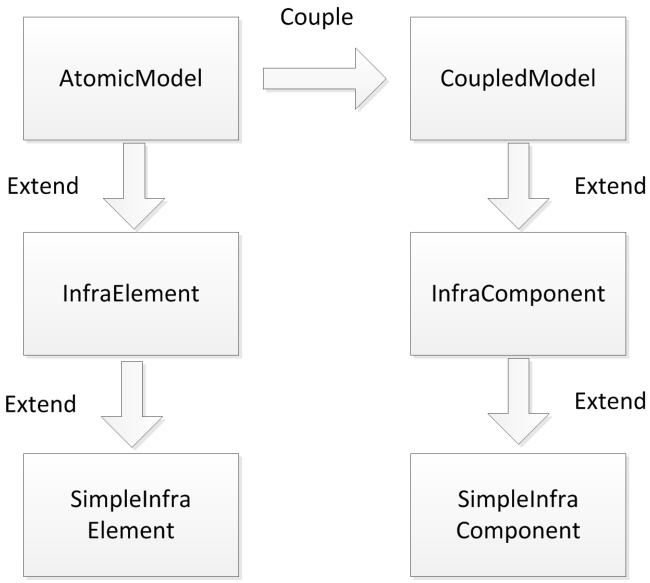


Figure B.1: Relationships between several classes

B.3 trackEntity

In *trackEntity*, *begpnt* and *endpnt* stores the information of the coordinates of the tracks read from the AutoCAD file. However, this is information is only used to draw the map of the model, which is not store in the entity of *TrackSegment*.

B.4 Depot

```
private void initDepot(String modelName) {
    inputFromControlCenter = new InputPort<MessageToDepot>(this);
    outputToControlCenter = new OutputPort<MessageToControlCenter>(this);
    depotControlUnit = new DepotControlUnit(this);

    this.addExternalInputCoupling(this.inputFromControlCenter,
            depotControlUnit.inputFromControlCenter);
    this.addExternalOutputCoupling(depotControlUnit.outputToControlCenter,
            this.outputToControlCenter);
}

public Exit connectToSource(String sourceName) {
    Exit source = new Exit(sourceName, this);
    this.addEndNode();
    this.linkNodes(source.getEndNode(), this.getEndNode(nrEndNodes++));
    this.addInternalCoupling(depotControlUnit.outputToSource,
            source.inputFromDCU);
    listOfSource.add(source);
    return source;
}

public Sink connectToSink() {
    Sink sink = new Sink(this);
    this.addStartNode();
    this.linkNodes(this.getStartNode(nrStartNodes++), sink.getStartNode());
    this.addInternalCoupling(sink.outputToControlUnit,
            depotControlUnit.inputFromSink);
    listOfSink.add(sink);
    return sink;
}
```

Figure B.2

```
protected void deltaExternal(double e, Object value) {

    if (this.activePort == inputFromSink) {
        nrVehicleAvailable++;
        this.sigma = Double.POSITIVE_INFINITY;
        return;
    }

    if (value instanceof MessageToDepot) // receive message from
                                         // ControlCenter
    {
        if (((MessageToDepot) value).nrVehicleToSend == 0) {
            sendMsgToControlCenter = true;
            sigma = 0;
        } else { // msg to dispatch new vehicle
            if (((Depot) this.getParentModel()) ==
                ((Depot) ((MessageToDepot) value).depot))
            {
                nrVehicleToDispatch =
                    ((MessageToDepot) value).nrVehicleToSend;
                vehicleInfo = ((MessageToDepot) value).vehicleInfo;
                dispatchNewVehicle = true;
                this.sigma = 0;
            } else
                sigma = Double.POSITIVE_INFINITY;
        }
    }
    // TODO Auto-generated method stub
}
```

Figure B.3

```
protected void lambda()
{

    if (sendMsgToControlCenter) {
        MessageToControlCenter msgToControlCenter
            = new MessageToControlCenter();
        msgToControlCenter.nrVehicleAvailable = nrStandbyVehicle;
        msgToControlCenter.sender = (Depot) this.getParentModel();
        msgToControlCenter.coordinates = this.coordinates;
        outputToControlCenter.send(msgToControlCenter);
        sendMsgToControlCenter = false;
        this.sigma = Double.POSITIVE_INFINITY;
        return;
    }

    if (dispatchNewVehicle) {

        MessageDCUToSource msgDCUToSource = new MessageDCUToSource(
                nrVehicleToDispatch,
                ((Depot) this.parentModel).depotName);
        msgDCUToSource.vehicleInfo = this.vehicleInfo;
        outputToSource.send(msgDCUToSource);
        decreaseNrStandbyVehicle();

        dispatchNewVehicle = false;
        this.sigma = Double.POSITIVE_INFINITY;
    }
}
```

Figure B.4

```
protected void deltaExternal(double e, Object value) {
    // TODO Auto-generated method stub
    if (value instanceof MessageDCUToSource) {
        if (this.parentModel.getModelName().equals(
                ((MessageDCUToSource) value).depotName)
                || (this.parentModel.getModelName())
                        .equals(((MessageDCUToSource) value).depotName
                                + "1")) {
            msg = (MessageDCUToSource) value;
            dispatchNewVehicle = true;
            this.sigma = 0;
        }
    }
}

protected void lambda() {
    if (dispatchNewVehicle) {
        double timeNow = 0;
        try {
            timeNow = simulator.getSimulatorTime();
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        TimeOfDay startTime = null;

        try {
            startTime = new TimeOfDay(new Date(this.simulator
                    .getReplication().getTreatment().getStartTime()));
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        int timeNow1 = (int) timeNow + 10
                + startTime.getTimeOfDayInSeconds();

        // set up information for the new vehicle
        TimeOfDay t = new TimeOfDay(timeNow1);
        Timetable timetable = new Timetable();
        timetable.substitute = true;
        timetable.vehicleInfo = msg.vehicleInfo;
        timetable.addScheduledTime(t);
        ((Exit) this.getParentModel()).addVehicleGenerator(timetable);

        sigma = Double.POSITIVE_INFINITY;
    }
}
```

Figure B.5

B.5 RailVehicle

In each integration time-step Δt, given the instantaneous acceleration rate of a vehicle at time t_n, it computes the vehicle's speed and position/distance at $t_{n+1} = t_n + \Delta t$ [37]. Movement of the vehicles are computed in the `computeMovement()`.

The computation of the movements gives the duration of the movement, the distance that the vehicle will travel during this movement, the end speed of this movement, and acceleration of the movement. Every time when a vehicle finishes one movement, it will first check if has another movement. If yes, it will execute the next movement; otherwise it will transit to the next phase, compute the movements for this phase and decide its next phase according to its state. When an external event occurs, the vehicle will update the information of the environment, for example the exit speed limit, distance to next infra or vehicle in the front, etc. Then the internal transition will be executed to compute the new movements of the vehicle.

When a vehicle is traveling along a `Exit`, a `StopBlock`, or an `Interlocking`, it is coupled as a child model of these coupled models. When a vehicle is traveling along the part where the track segment, the vehicle is directly internally coupled with this track segment. In this way, the messages sent along the infrastructure can be received by the vehicle and the messages from the vehicles can be propagated by the infrastructure.

A rail vehicle has different *phase*s, including `MOVE_TO_NEXT_TRACK`, `START`, `FOLLOW`, `STOP`, `HALTING` and `SCHEDULED_WAITING`. All the behaviors of the vehicles can be categorized into the combinations of these phases. Each time when a vehicle changes into a new phase, it calculates its new *movements*, which can be composed of acceleration, moving with constant speed, deceleration, and halting. The *movements* are computed according to the environment of the vehicle and its own state. The environment of the vehicle consists the distance to the next infrastructure, the speed limit of the next infrastructure, whether there is a stop in the front, what is the distance between this vehicle and the vehicle in front, etc. The state of the vehicle can be its *phase* and current speed.

When a vehicle is trying to report disturbance to the control center, it needs to "find" the control center first in the model.

```
public String getVehicleIDForInterlocking()
{
    if (substitute){
            return "0+" + this.substituteLineBeginning;
    }
    else if (!deadhead)
        return getVehicleID();
    else
        return getVehicleID() + "+" + this.lineBeginning;
}
```

Figure B.6: GetVehicleIDForInterlocking

```
if (!stop.getFullName().equals(previousStopID)) {
    // update the name of the last stop and delay time
    previousStop = stopName;
    previousStopID = stop.getFullName();

    if (!deadhead && !substitute) {
        // delay is only calculated when the vehicle is on service
        // trip
        if (stop.tripID.get(lineID) != null) {

            int i = stop.tripID.get(lineID);
            i++;
            stop.tripID.remove(lineID);
            stop.tripID.put(lineID, i);
        } else
            stop.tripID.put(lineID, 1);
        this.currentTripID = stop.tripID.get(lineID);

        if (stop.getScheduledDepartureTime1(lineID,
                this.currentTripID) != null) {
            TimeOfDay tStop = stop.getScheduledDepartureTime1(
                    lineID, this.currentTripID);
            TimeOfDay currentTime = getCurrentTimeOfDay();

            if (tStop != null) {
                if (currentTime.deviationInSecond(tStop) > 0)
                    arrivalTimeDifference = currentTime
                            .deviationInSecond(tStop);
                else
                    arrivalTimeDifference = 0;

                delay = currentTime.deviationInSecond(tStop);
                if (stopNameAreEqual(stopName, this.lineBeginning)
                        && delay < 0)
                    // the beginning of the line
                    delay = 0;
                if (stopNameAreEqual(stopName, this.lineEnd + " e")
                        && !this.substitute && !this.deadhead) {
                    // just enter into the last
                    tripDuration = getCurrentTimeOfDay()
                            .deviationInSecond(tripStartTime);
                }
                delayTime += Math.max(delay, 0);
                delayCounter++;
            }
        }
    }
```

Figure B.7

```
if(stopNameAreEqual(stopName, this.lineBeginning) ){ //the beginning of the line

    deadhead = false;
    if (DoubleCompare.greaterThan(tripDistance, 0)){
        tripNr++;
        changeTripDirection();
    }
    tripDistance = 0;
    tripDuration = 0;

}else if(stopNameAreEqual(stopName, this.lineEnd + " e")
        && !this.substitute && !this.deadhead){//just enter into the last
    atLineEnd = true;
}
```

Figure B.8: Set Deadhead

```
if (this.deadhead) {
    totalDeadheadTravelled +=
        (totalDistanceToNxtInfra - distanceToNxtInfra);
} else
    tripDistance +=
        (totalDistanceToNxtInfra - distanceToNxtInfra);
```

Figure B.9: Deadhead kilometer calculation

```
if (isToHalt())
{
    calculateDwellTime(); //van
    movement.put(new double[] {
            Math.max(dwellTime + extraDwellTime, waitingTime), 0, 0,
            0 });
    myPhase = MyPhase.HALTING;
    sigma = Math.max(dwellTime + extraDwellTime, waitingTime);
} else
```

Figure B.10: If the vehicle is to halt at the stop

```
private void calculateDwellTime(){
    if(((InfraInterface)this.parentModel).getModelType()
            == ModelType.STOP){
        Stop stop = (Stop)this.getParentModel();
        String stopName = stop.getName();
        dwellTime = getDwellTimeFromStop();

        if (stopNameAreEqual(stopName, this.lineBeginning)) {
            // the beginning of the line

            this.startingPoint = this.lineBeginning;
            this.nextCheckingPoint = this.lineBeginning;
            this.lineBeginning = "nothing";

            TimeOfDay tStop = null;
            TimeOfDay currentTime = null;

            if (substitute) {
                this.dwellTime = 0;
                this.extraDwellTime = 0;
            } else if (stop.getScheduledDepartureTime(lineID,
                    this.currentTripID) != null) {
                tStop = stop.getScheduledDepartureTime(lineID,
                        this.currentTripID);
                currentTime = getCurrentTimeOfDay();
                extraDwellTime = -currentTime.deviationInSecond(tStop);
            } else {
                tStop = stop.getScheduledDepartureTime1(lineID,
                        this.currentTripID);
                currentTime = getCurrentTimeOfDay();
                if (tStop != null)
                    extraDwellTime =
                        -currentTime.deviationInSecond(tStop);
            }

            if (extraDwellTime >= 0)
                this.tripStartTime = tStop;
            else
                tripStartTime = currentTime;

            if (extraDwellTime > dwellTime)
                extraDwellTime = extraDwellTime - dwellTime;
            else
                extraDwellTime = 0;
        }
        else
            extraDwellTime = 0;
    }
}
```

Figure B.11: Calculate dwell time

```
else if (this.nxtInfra.getModelType() == ModelType.TRACK_SEGMENT
        && ((InfraElement) nxtInfra).getSpeedLimit() == 0) {

    if (((TrackSegment) this.nxtInfra).isDisturbanceMode()) {
        CoupledModel parentModel = this.getParentModel();
        TrackSegment t = (TrackSegment) this
                .getOccupiedInfraElements().peekLast();

        reportDisturbanceToControlCenter((TrackSegment) this.nxtInfra);

        parentModel.removeInternalCoupling(
                this.getOutputForward(),
                t.getInputForward());
        parentModel.removeInternalCoupling(
                this.getOutputBackward(),
                t.getInputBackward());
        parentModel.removeInternalCoupling(
                t.getOutputToVehicle(),
                this.getInputFromTrack());
        ((TrackSegment) this.getOccupiedInfraElements()
                .peekLast()).clearObjectsOnMe();
        this.parentModel.removeModelComponent(this);
        System.out.println("==REMOVE " + this
                + " AT PROBLEMATIC TRACK==" + t.toString());

        this.sigma = Double.POSITIVE_INFINITY;
        return;
    } else {
        connectToControlCenter();
        this.clearVehicleAhead();
        this.sigma = Double.POSITIVE_INFINITY;
    }

    if (restart) {
        myPhase = MyPhase.START;
        restart = false;
        sigma = 0;
    }
}
```

Figure B.12: In disturbance

B.6 VehicleGenerator

The internal transition function of `VehicleGenerators` is triggered according to this timetable. Every time after generating one vehicle, the time advance for the *VehicleGenerator* is calculated as the time difference until the time to dispatch next vehicle, and the internal transition is scheduled.

B.7 DataOutputer

In package *nl.tudelft.simulation.libros.io*. This class is used to define the data into the database file. The SQL statement to create the tables is stored in the file called *tables.sql*. In `DataOutputer`, the SQL statement is prepared for inserting data into the database each time.

B.8 LibrosModel

Some global parameters are defined in this class. Animation, back ground plot and data output function can be switched on or off by specifying the corresponding booleans variables. The path and file name of the input files, including the AutoCAD file and Excel files are also specified here.

B.9 CadModelBuilder

The orientation of the tracks are defined in *regulateConnectedTrackOrientation*.

```
else {
    Depot depot = generateAndConfigDepot(sourceEntity);
    generateModelTree(sourceEntity,
            depot.getEndNode(depot.nrEndNodes - 1));
    depot.depotControlUnit
            .setCoordinates(getPoint3d(getBeginPoint(nxtEntity)));
}
```

Figure B.13: Connect depot with the network

```
private Depot findOrCreateDepot(String name)
{
    Depot depot = null;
    for (Depot d : listOfDepot) {
        if (name.startsWith(d.depotName)
                && name.length() < d.depotName.length() + 2) {
            depot = d;
            break;
        }
    }
    String depotName = getDepotName(name);

    if (depot == null) {
        depot = new Depot(depotName, this.tlm);

        this.tlm.addInternalCoupling(depot.outputToControlCenter,
                this.tlm.getControlCenter().inputFromDepot);
        this.tlm.addInternalCoupling(
                this.tlm.getControlCenter().outputToDepot,
                depot.inputFromControlCenter);
        listOfDepot.add(depot);
    }

    return depot;
}
```

Figure B.14: Find or create depot

```
if (getEntityACI(entity) == ACI.LightGreen.colorIndex()) {
    for (int j = 0; j < track.size(); j++) {
        disturbanceTrackEntities.add(track.get(j));
    }
}
```

Figure B.15: Disturbance track

```
private YxxfEntInsert findSink(YxxfEnt trackEntity)
        throws ModelGenerationException
{
    YxxfGfxPointW refPnt = getEndPoint(trackEntity);

    for (YxxfEntInsert sinkEntity : sinkEntities) {
        if (areConnected(refPnt, sinkEntity, SNAPPING_TOLERANCE)) {
            return sinkEntity;
        }
    }
    return null;
}
```

Figure B.16: Fink sink

B.10 Constructing speed limit

In LIBROS-II, the vehicles drive according to the speed limit of the infrastructure. In order to determine the speed limit for all the infrastructures, the model follows several steps. First, the general speed limit of the infrastructure according to their location in the network according to the color of the tracks defined in the AutoCAD file, which can be classified into categories: *StreetTrack*, *FreeLane* and *CityTrack* Second, the speed limit is also limited by if the infrastructure is in a stop or an intersection. Their speed limits are usually lower than normal tracks. Third, the speed limit is also regulated according to the length of the track and speed limit of the following infrastructure. It is adjusted in a way that, with the deceleration rate, the vehicle can reduce its speed to the speed limit of the next infrastructure before the end of this infrastructure.

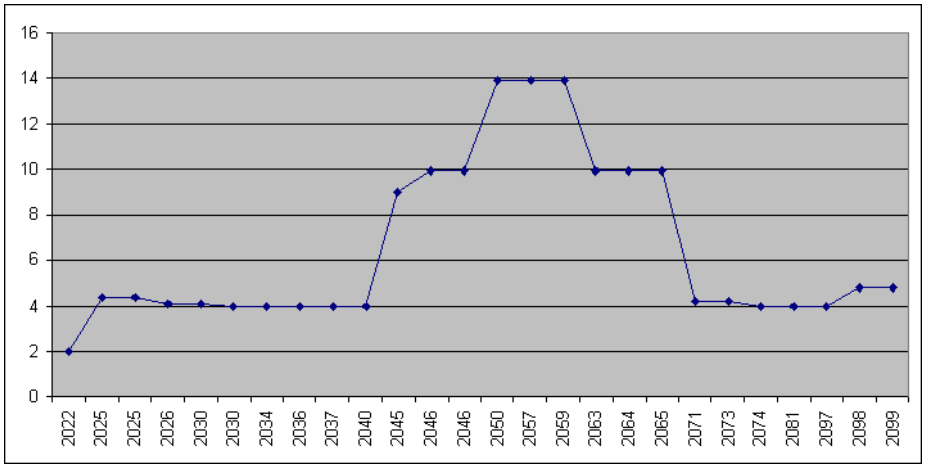


Figure B.17: Plot of speed of a vehicle in the model traveling from “Zwarte Pad” to “Kurhaus”

From comparing the result from the model and the real data, we concluded that the vehicles in the model drive much slower than those in reality. For example, in the model, the vehicles of line 1 driving from “Zwarte Pad” to stop “Kurhaus” takes around 80 seconds, but in the service timetable provided by HTM, the difference between the two stops is one minute (notice that the service table uses minute as the basic unit, so around 1 minute will all the rounded up to 1 minute). What’s more, from the result of the model, in every stop the delay time is increasing. This shows that the vehicles are constantly delays and is driving slower than the vehicles in reality in general. After the adjustment of method *checkSpeedLimit()* in `CadModel`, the overall speed of the vehicles has been improved to the same level as the vehicles in reality.

There is in fact a difference between drivers’ behavior in reality and what is regulated in the speed limit for safety reasons. First, the halting time of the vehicles at each stop is defined as... However, for real vehicles, they normally stop for a period time shorter than this. Especially when there are less passengers, the drivers do not wait for a long time for nobody. Second, there is strict rules on the speed limit of the vehicle on the curve tracks: the smaller the radius of the curve track... However, because the speed limit of some curvy tracks is quite small, normally drivers do not reduce the speed really to this level. Third, the speed limit for all the intersections is 4 m/s, but the drivers do not follow this. From figure B.17, although the speed limit is normally

around 16, most of the time the speed limit is much lower than 16.

B.11 Special attention with programming with LIBROS-II

There are certain types of warnings that you can easily get if you want to program with LIBROS-II.

`sigma` is the time advance of transition function, should never be changed outside of the transitions functions. `sigma` is used every time to schedule the next event. Every time when a model receives an external event through the input port, it will compare the elapsed time with the sigma of the model. If the elapsed time is smaller or equal to the sigma, this is normal then the program will go on. If the elapsed time is bigger then the sigma, it means that there is something wrong in the transition functions. As a result, sigma `sigma` should never be changed outside of the transition functions.

During debugging, if warning shows that in external transition, the elapsed time is bigger than `sigma` (time advance). This could happen due to some reasons. First, `sigma` is changed some where outside the transition functions. As a result, the new "`sigma`" could be smaller than what was used as the time advance, thus causing this problem. Second, during the computation of internal transition, as some codes could not be executed, this model will jump over the entire transition function, thus leaving `sigma` with a wrong value, which will result in this warning. As a result, we should go back to previous phase of the same atomic model (could be `RailVehicle` or InterlockingControlUnit), and examine the detail computation steps of the model.

Before you are sure that the model is correct, do not bring randomness into the model yet. This will be helpful for the debugging process to track back to the problematic location.

B.12 Input files

B.12.1 Vehicle Schedule

B.12.2 Optimization model solver

The optimization solver takes the distance between each end of the line and each depot as input, number of trips from each direction and capacity of the depots as constraints, minimizing total deadhead-kilometer as objective function, as number of vehicles for each line parking in each depot as variables. Because even with the same design of the depots, deadhead-kilometer can differ a lot due to different vehicle plannings, this solver gives a common ground to the comparison among alternatives.

B.12.3 deadheadRoute.xls

All the deadhead routes are defined in this file. The deadhead routes are defined with both the number of the line and the destination. If it is a normal deadhead trip, then the destination is the name of the first stop of the line. If it is trip of the substitute vehicle, then the number of the line is used as "0", and the the destination is the name of the last stop that the disturbance vehicle passes before the disturbance.

B.12.4 initialValue.xls

This file stores the vehicle allocation among the depots of each line. Basically it is the output of the optimization solver. The file also includes the number of the standby vehicles in each depot.

```
public HashMap<String, LinkedList<TimeOfDay>> searchVehicleSchedule(
        Exit source) {
    if (source.initialValue.size() != 0) {
        HashMap<String, LinkedList<TimeOfDay>> vehicleSchedule =
            new HashMap<String, LinkedList<TimeOfDay>>();
        for (int i = 0; i < source.lines.size(); i++) {
            resetSearchPatterns();
            addSearchPattern(lineNameColName1, source.lines.get(i));
            ArrayList<Row> timetableRows = findAllRows();
            for (Row row : timetableRows) {
                String lineName = row.getCell(
                        CellReference
                                .convertColStringToIndex(lineNameColName1))
                        .toString();
                lineName = toIntForm(lineName);
                for (Cell cell : row) {
                    if (cell != null
                            && cell.getCellType() == Cell.CELL_TYPE_NUMERIC
                            && DateUtil.isCellDateFormatted(cell)) {
                        TimeOfDay tod = new TimeOfDay(getCell(
                                cell.getRowIndex() + 1,
                                cell.getColumnIndex()).getDateCellValue());
                        if (tod.isAfter(new TimeOfDay(4, 0, 0))) {
                            if (getCell(cell.getRowIndex() - 1,
                                    cell.getColumnIndex()).toString() != null) {
                                String sequenceNr = getCell(
                                        cell.getRowIndex() - 1,
                                        cell.getColumnIndex()).toString();
                                sequenceNr = toIntForm(sequenceNr);
                                if (vehicleSchedule.containsKey(sequenceNr)) {
                                    LinkedList<TimeOfDay> list =
                                        (LinkedList<TimeOfDay>) vehicleSchedule
                                            .get(sequenceNr);
                                    sortAddDate(list, tod);
                                } else {
                                    LinkedList<TimeOfDay> list =
                                        new LinkedList<TimeOfDay>();
                                    list.add(tod);
                                    vehicleSchedule.put(sequenceNr, list);
                                }
                            }
                        }
                    }
                }
            }
        }
        source.vehicleSchedule = vehicleSchedule;
        return vehicleSchedule;
    }
    return null;
}
```

Figure B.18

```
public class DeadheadRouteReader extends XlsReader {
    public static int lineIDrowIndex = 0;
    public static int startingPointRowIndex = 1;
    public static int lineBeginningRowIndex = 2;
    public static final String depotcolName = "C";

    public DeadheadRouteReader(String filePath) throws IOException {
        readXls(filePath);
        resetSearchPatterns();
    }

    public HashMap<String, PointPosition> searchPointConfig(String stopID) {
        HashMap<String, PointPosition> pointConfigDeadhead =
            new HashMap<String, PointPosition>();

        if (stopID != "") {
            ArrayList<Cell> cells = findAllCells(stopID);

            if (cells.size() == 0) {
                System.err.println("no fp ID in the file");
            }

            for (Cell cell : cells) {
                String lineID = cellContentToString(getCell(lineIDrowIndex,
                        cell.getColumnIndex()));
                String startingPoint = cellContentToString(getCell(
                        startingPointRowIndex, cell.getColumnIndex()));
                String lineBeginning = cellContentToString(getCell(
                        lineBeginningRowIndex, cell.getColumnIndex()));
                String lindAndStartingPoint = lineID + "+" + lineBeginning;
                String pointPos = cellContentToString(getCell(
                        cell.getRowIndex(), cell.getColumnIndex() + 1));
                if (pointPos.equalsIgnoreCase("L")) {
                    pointConfigDeadhead.put(lindAndStartingPoint,
                            PointPosition.LEFT);
                } else if (pointPos.equalsIgnoreCase("R")) {
                    pointConfigDeadhead.put(lindAndStartingPoint,
                            PointPosition.RIGHT);
                }
            }
        }
        return pointConfigDeadhead;
    }
}
```

Figure B.19

```
public class InitialValueReader
extends XlsReader
{
    public static final String depotColName = "A";
    public static final String iniNrStandbyVehicleColName = "B";
    public static final String vehicleColorColName = "C";

    public InitialValueReader(String filePath) throws IOException
    {
        readXls(filePath);
    }

    public void getInitialValue(Exit source) {
        resetSearchPatterns();

        addSearchPattern(depotColName, source.getModelName());
        int nrVehicle = 0;

        Row row = findRow();
        if (row != null) {

            for (Cell cell : row) {
                if (cell.getCellType() == Cell.CELL_TYPE_NUMERIC) {
                    String s = cell.toString();
                    int i = s.indexOf(".");
                    String sj = s.substring(0, i);
                    nrVehicle = Integer.valueOf(sj);
                    String lineAndStop = getCell(cell.getRowIndex() - 1,
                            cell.getColumnIndex()).toString();
                    if (!lineAndStop.equals("nrStandbyVehicle")) {
                        source.initialValue.add(new StopAndNrVehicle(
                                lineAndStop, nrVehicle));
                        if (lineAndStop.contains("+")) {
                            String line = lineAndStop.substring(0,
                                    lineAndStop.indexOf("+"));
                            if (!source.lines.contains(line))
                                source.lines.add(line);
                        }
                    } else
                        ((Depot) source.getParentModel()).depotControlUnit
                                .initializeNrStandbyVehicle(nrVehicle);
                }

            }
        }
```

Figure B.20

C Design

Figure C.1: Depot Zichtenburg entrance and exit 1

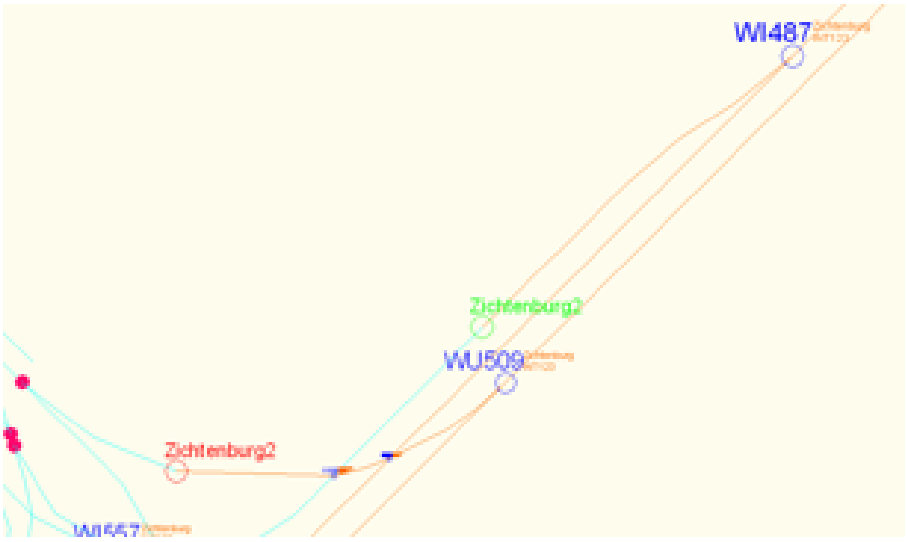


Figure C.2: Depot Zichtenburg entrance and exit 2

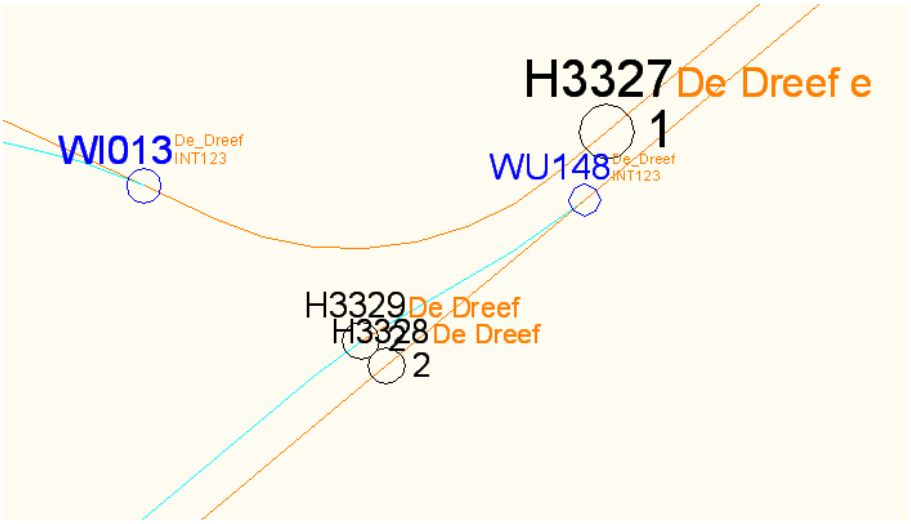


Figure C.3: End stop of the line with an extra “e” to its name

D Verification and Validation

D.1 Validation: Trip Distance

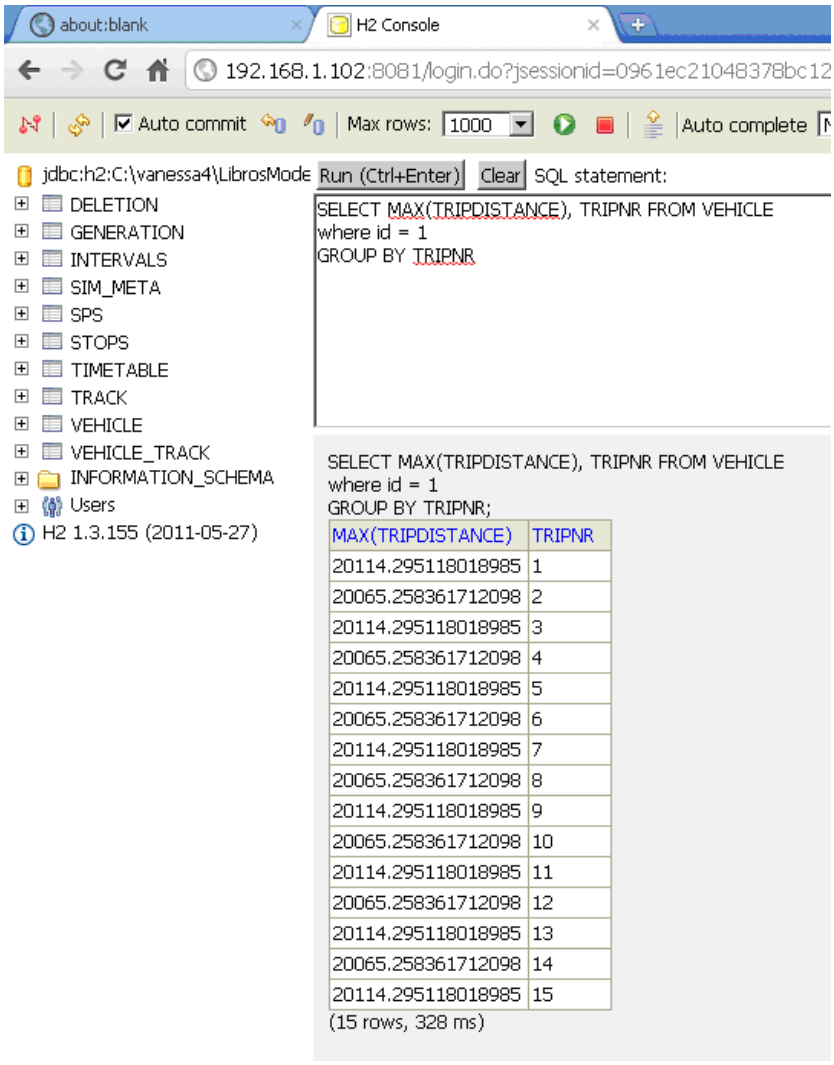


Figure D.1: Result of trip distance from the model of two directions of line 1

D.2 Validation: Statistical Test

Here we can use the two-sample Student's t-test. This is because we do not know the standard deviation of the population and we want to compare the means of two samples. Before the test, we should make sure that the statistical of the test follows a normal distribution. According to Central Limit Theorem, if the population from which you sample is extremely non-normal, the sampling distribution of the mean will still be approximately normal given a large enough sample size (some author suggest for sample sizes of 300 or greater). Therefore, we use a sample size of greater than 300 for each direction in this test.

The Null hypothesis is that the difference between the deadhead-kilometer of the two alternatives is 0. So the null hypothesis is that the difference is not 0.

So the test statistic

$$T = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{\frac{{SD_1}^2}{\eta_1} + \frac{{SD_2}^2}{\eta_2}}}$$

Where

$\overline{x_1}$ is the sample mean of alternative current situation,
$\overline{x_2}$ is the mean of alternative with remise zuid,
SD_1 is the standard deviation of sample of alternative current situation,
SD_2 is the standard deviation of sample of alternative with remise zuid,
η_1 is the sample size of alternative current situation and
η_2 is the sample size of alternative with remise zuid.

$$T = 68$$

The critical value, with 90% confidence level and degree of freedom $(\eta_1 - 1) + (\eta_2 - 1)$ is $t_{0.1/2,6} = 1.94$.
Since

$$T > t_{0.05,6}$$

so that we can reject the null hypothesis, which means that we know that the difference between the means is significantly different from 0. The results are shown in tableD.1. An example of the test statistics with SPSS is shown in figure D.2.

Line and direction	P-value	Significantly different
Line 1 direction 1	0.000	yes
Line 1 direction 2	0.000	yes
Line 6 direction 1	0.000	yes
Line 6 direction 2	0.000	yes
Line 9 direction 1	0.000	yes
Line 9 direction 2	0.000	yes
Line 17 direction 1	0.000	yes
Line 17 direction 2	0.000	yes

Table D.1: Results of t-test for service trip durations

T-Test

[DataSet0]

One-Sample Statistics

	N	Mean	Std. Deviation	Std. Error Mean
VAR00001	299	3504.5518	35.45041	2.05015

One-Sample Test

	Test Value = 3742.56					
					90% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
VAR00001	-116.093	298	.000	-238.00816	-241.3909	-234.6254

Figure D.2: Example of t-test result of line 1 direction 1

E More Results of the Analysis of the Impacts

E.1 Detailed Results of the Analysis

1	Lijncode										
	Techniek										
11000	Weekdagen										
1	Richtingnr										
	Richting										
1	212	Zwarte pad		5:04		5:23		5:38		5:53	6:01
1	202	Kurhaus		5:05		5:24		5:39		5:54	6:02
1	706	Scheveningseslag		5:06		5:25		5:40		5:55	6:03
1	702	Badhuiskade		5:07		5:26		5:41		5:56	6:04
1	714	Keizerstraat		5:08		5:27		5:42		5:57	6:05
1	722	Duinstraat		5:09		5:28		5:43		5:58	6:06
1	907	Frankenslag		5:10		5:29		5:44		5:59	6:08
1	922	World Forum		5:11		5:30		5:45		6:00	6:09
1	1001	Adriaan Goekooplaan		5:12		5:31		5:46		6:01	6:10
1	501	Ary van der Spuyweg		5:13		5:32		5:47		6:02	6:11
1	508	Vredespaleis		5:14		5:33		5:48		6:03	6:12
1	2301	Javastraat		5:15		5:34		5:49		6:04	6:13
1	2308	Mauritskade		5:16		5:35		5:50		6:05	6:15
1	2804	Kneuterdijk		5:17		5:36		5:51		6:06	6:16
1	2832	Centrum		5:19		5:38		5:53		6:08	6:18
1	2851	Bierkade		5:21		5:40		5:55		6:10	6:20
1	2730	Station Hollands Spoor		5:23		5:42		5:57		6:12	6:23
1	3840	Leeghwaterplein	5:20	5:25	5:40	5:44	5:55	5:59	6:10	6:14	6:25
1	3810	Oudemansstraat	5:21	5:26	5:41	5:45	5:56	6:00	6:11	6:15	6:26
1	3824	Lorentzplein	5:22	5:27	5:42	5:46	5:57	6:01	6:12	6:16	6:27
1	3838	Broeksloot	5:24	5:29	5:44	5:48	5:59	6:03	6:14	6:18	6:28
1	5104	Herenstraat	5:25	5:30	5:45	5:49	6:00	6:04	6:15	6:19	6:30
1	5107	Hoornbrug	5:26	5:31	5:46	5:50	6:01	6:05	6:16	6:20	6:31
1	5901	Broekpolder	5:27	5:32	5:47	5:51	6:02	6:06	6:17	6:21	6:32
1	5905	s-Gravenmade	5:28	5:33	5:48	5:52	6:03	6:07	6:18	6:22	6:33
1	5911	Vlietbrug	5:29	5:34	5:49	5:53	6:04	6:08	6:19	6:23	6:33
1	5907	Verffabriek	5:31	5:36	5:51	5:55	6:06	6:10	6:21	6:25	6:34
1	9003	Brasserskade	5:34	5:39	5:54	5:58	6:09	6:13	6:24	6:28	6:37
1	9005	Nieuwe Plantage	5:35	5:40	5:55	5:59	6:10	6:14	6:25	6:29	6:39
1	9007	Wateringsevest	5:36	5:41	5:56	6:00	6:11	6:15	6:26	6:30	6:40
1	9009	Prinsenhof	5:37	5:42	5:57	6:01	6:12	6:16	6:27	6:31	6:41
1	9017	Delft Station	5:39	5:45	5:59	6:04	6:14	6:19	6:29	6:34	6:43
1	9101	Krakeelpolderweg	5:40	5:46	6:00	6:05	6:15	6:20	6:30	6:35	6:45
1	9103	Hovenpassage	5:42	5:47	6:02	6:06	6:17	6:21	6:32	6:36	6:46

Figure E.1: Service timetable of line 1 from Zwarte Pad (red ones are not included in the analysis, and blue ones are ignored when adjusting the service timetable for the future)

Direction	1	2
1	Zwarte Pad	Abtswoudsepark
6	Leyeburg	MCH Antoniushove
9	Zwarte Pad	De Dreef
17	Frankenslag	Dorpskade

Table E.1: First stops of the directions of the lines

1	Lijncode							
	Techniek							
1111000	Weekdagen							
2	Richtingnr							
rand	Richting							
1	9117	Abtswoudsepark	6:00	6:19	6:34		6:49	
1	9114	Bikolaan	6:02	6:21	6:36		6:51	
1	9112	Sadatweg	6:03	6:22	6:37		6:52	
1	9110	Van der Slootsingel	6:05	6:24	6:39		6:54	
1	9108	Diepenbrockstraat	6:06	6:25	6:40		6:55	
1	9106	Aart van der Leeuwlaan	6:07	6:26	6:41		6:56	
1	9104	Hovenpassage	6:08	6:27	6:42		6:57	
1	9102	Krakeelpolderweg	6:10	6:29	6:44		6:59	
1	9018	Delft Station	6:11	6:30	6:45		7:00	
1	9010	Prinsenhof	6:13	6:32	6:47		7:02	
1	9008	Wateringsevest	6:15	6:34	6:49		7:04	
1	9006	Nieuwe Plantage	6:16	6:35	6:50		7:05	
1	9004	Brasserskade	6:17	6:36	6:51		7:06	
1	5908	Verffabriek	6:19	6:38	6:53		7:08	
1	5912	Vlietbrug	6:20	6:39	6:54		7:09	
1	5906	s-Gravenmade	6:21	6:40	6:55		7:10	
1	5902	Broekpolder	6:22	6:41	6:56		7:11	
1	5108	Hoornbrug	6:23	6:42	6:57		7:12	
1	5103	Herenstraat	6:24	6:43	6:58		7:13	
1	3815	Broeksloot	6:26	6:45	7:00		7:15	
1	3823	Lorentzplein	6:27	6:46	7:01		7:16	
1	3809	Oudemansstraat	6:28	6:47	7:02		7:17	
1	3839	Leeghwaterplein	6:29	6:48	7:03		7:18	
1	2731	Station Hollands Spoor	6:31	6:50	7:05		7:20	
1	2852	Bierkade	6:34	6:53	7:08		7:23	
1	2833	Centrum	6:36	6:55	7:10		7:25	
1	2803	Kneuterdijk	6:38	6:57	7:12		7:27	
1	2307	Mauritskade	6:39	6:58	7:13		7:28	
1	2302	Javastraat	6:40	6:59	7:14		7:29	
1	509	Vredespaleis	6:41	7:00	7:15		7:30	
1	502	Ary van der Spuyweg	6:42	7:01	7:16		7:31	
1	1002	Adriaan Goekooplaan	6:43	7:02	7:17		7:32	
1	921	World Forum	6:44	7:03	7:18		7:33	
1	906	Frankenslag	6:46	7:05	7:20		7:35	

Figure E.2: Service timetable of line 1 from Abtswoudsepark

lijn	richting	Lijsterbes	Scheveningen	Zichtenburg	Leidsch. NS	Oosterheem	in/uit		
1	Scheveningen	0	10	0	0	0	10	10	GTL
1	Delft	4	0	0	0	0	4	4	GTL
2	Kraaijenstein	0	0	8	0	0	8	8	GTL
2	Leidschendam	6	0	0	0	0	6	6	GTL
3	Loosduinen	0	0	7	0	0	7	7	RR
3	Zoetermeer Centrum	0	0	0	11	0	11	11	RR
3K	Lohmanplein	0	0	2	0	0	2	2	RR
3K	Zoetermeer Centrum	0	0	0	2	0	2	2	RR
4	Uithof	0	0	6	0	0	6	6	RR
4	Javalaan	0	0	0	1	8	9	9	RR
4K	Monstersestraat	0	0	4	0	0	4	4	RR
4K	Javalaan	0	0	0	8	0	8	8	RR
6	Leijenburg	0	0	9	0	0	9	9	GTL
6	Leidsch. Leidschenhage	0	0	0	0	0	0	0	GTL
9K	Madurodam	0	2	0	0	0	2	2	GTL
9K	Vrederust	0	0	2	0	0	2	2	GTL
9	Scheveningen	0	13	0	0	0	13	13	GTL
9	Vrederust	0	0	2	0	0	2	2	GTL
10	Statenkwartier	0	6	0	0	0	6	6	GTL
10	Voorburg Station	0	0	0	0	0	0	0	GTL
11	Rijswijkseplein	3	0	0	0	0	3	3	GTL
11	Scheveningen Haven	0	4	0	0	0	4	4	GTL
12	Duindorp	4	0	0	0	0	4	4	GTL
12	Rijswijkseplein	5	0	0	0	0	5	5	GTL
15	Nootdorp	8	0	0	0	0	8	8	GTL
16	Wateringen	0	0	9	0	0	9	9	GTL
17	Statenkwartier	0	12	0	0	0	12	12	GTL
17	Wateringen	0	0	2	0	0	2	2	GTL
19	Delft	5	0	0	0	0	5	5	GTL
19	Leidsch. Leidschenhage	2	0	0	0	0	2	2	GTL
	GTL	**37**	**47**	**32**	**0**	**0**	**116**	116	
	RR	**0**	**0**	**19**	**22**	**8**	**49**		
	NBT	**0**	**0**	**0**	**0**	**0**	**0**		
	BREED	**0**	**0**	**19**	**22**	**8**	**49**	49	
	capaciteit smal	42	70	52			164		
	capaciteit breed			24	31	8	63		

Figure E.3: Vehicle allocation from the optimization model for alternative “Current situation”

lijn	richting	Zwarte Pad	Remise Zuid	Zichtenburg	Leidsch. NS	Oosterheem	in/uit		
1	Scheveningen	0	0	10	0	0	10	10	GTL
1	Delft	0	4	0	0	0	4	4	GTL
2	Kraaijenstein	0	0	8	0	0	8	8	RR
2	Leidschendam	0	0	6	0	0	6	6	RR
3	Loosduinen	0	0	0	7	0	7	7	RR
3	Zoetermeer Centrum	0	0	0	11	0	11	11	RR
3K	Lohmanplein	0	0	0	2	0	2	2	RR
3K	Zoetermeer Centrum	0	0	0	2	0	2	2	RR
4	Uithof	0	0	6	0	0	6	6	RR
4	Javalaan	0	0	0	0	9	9	9	RR
4K	Monstersestraat	0	0	0	4	0	4	4	RR
4K	Javalaan	0	0	0	0	8	8	8	RR
6	Leijenburg	0	0	9	0	0	9	9	GTL
6	Leidsch. Leidschenhage	0	0	0	0	0	0	0	GTL
9K	Madurodam	0	0	0	0	0	0	0	NBT
9K	Vrederust	0	0	4	0	0	4	4	NBT
9	Scheveningen	13	0	0	0	0	13	13	NBT
9	Vrederust	0	0	2	0	0	2	2	NBT
10	Statenkwartier	0	0	0	0	0	0		GTL
10	Voorburg Station	0	0	0	0	0	0		GTL
11	Rijswijkseplein	0	3	0	0	0	3	3	NBT
11	Scheveningen Haven	0	0	4	0	0	4	4	NBT
12	Duindorp	0	0	4	0	0	4	4	GTL
12	Rijswijkseplein	0	4	1	0	0	5	5	GTL
15	Nootdorp	0	8	0	0	0	8	8	GTL
16	Wateringen	0	0	9	0	0	9	9	GTL
17	Statenkwartier	11	0	1	0	0	12	12	NBT
17	Wateringen	0	0	2	0	0	2	2	NBT
19	Delft	0	5	0	0	0	5	5	GTL
19	Leidsch. Leidschenhage	0	2	0	0	0	2	2	GTL
	GTL	**0**	**23**	**33**	**0**	**0**	**56**	110	
100	**RR**	**0**	**0**	**20**	**26**	**17**	**63**		
	NBT	**24**	**3**	**13**	**0**	**0**	**40**		
	BREED	**24**	**3**	**33**	**26**	**17**	**103**	49	
	Breed	**24**	**20**	**58**	**26**	**17**			
	Smal			**77**					
	capaciteit smal	0					0		
	capaciteit breed	24	40	58	31	17	170		

Figure E.4: Vehicle allocation from the optimization model for alternative "With Remise Zuid"

lijn	richting	Zwarte Pad	Remise Zuid	Zichtenburg	Leidsch. NS	Oosterheem	in/uit		
1	Scheveningen	0	0	10	0	0	10	10	GTL
1	Delft	0	0	4	0	0	4	4	GTL
2	Kraaijenstein	0	0	8	0	0	8	8	RR
2	Leidschendam	0	0	6	0	0	6	6	RR
3	Loosduinen	0	0	0	7	0	7	7	RR
3	Zoetermeer Centrum	0	0	0	11	0	11	11	RR
3K	Lohmanplein	0	0	0	2	0	2	2	RR
3K	Zoetermeer Centrum	0	0	0	2	0	2	2	RR
4	Uithof	0	0	2	4	0	6	6	RR
4	Javalaan	0	0	0	0	9	9	9	RR
4K	Monstersestraat	0	0	0	4	0	4	4	RR
4K	Javalaan	0	0	0	0	8	8	8	RR
6	Leijenburg	0	0	9	0	0	9	9	GTL
6	Leidsch. Leidschenhage	0	0	0	0	0	0	0	GTL
9K	Madurodam	0	0	0	0	0	0	0	NBT
9K	Vrederust	4	0	0	0	0	4	4	NBT
9	Scheveningen	13	0	0	0	0	13	13	NBT
9	Vrederust	2	0	0	0	0	2	2	NBT
10	Statenkwartier	0	0	0	0	0	0		GTL
10	Voorburg Station	0	0	0	0	0	0		GTL
11	Rijswijkseplein	3	0	0	0	0	3	3	NBT
11	Scheveningen Haven	4	0	0	0	0	4	4	NBT
12	Duindorp	0	0	4	0	0	4	4	GTL
12	Rijswijkseplein	0	0	5	0	0	5	5	GTL
15	Nootdorp	0	0	8	0	0	8	8	GTL
16	Wateringen	0	0	9	0	0	9	9	GTL
17	Statenkwartier	12	0	0	0	0	12	12	NBT
17	Wateringen	2	0	0	0	0	2	2	NBT
19	Delft	0	0	5	0	0	5	5	GTL
19	Leidsch. Leidschenhage	0	0	2	0	0	2	2	GTL
	GTL	**0**	**0**	**56**	**0**	**0**	**56**	110	
100	**RR**	**0**	**0**	**16**	**30**	**17**	**63**		
	NBT	**40**	**0**	**0**	**0**	**0**	**40**		
	BREED	**40**	**0**	**16**	**30**	**17**	**103**	49	
	Breed	**40**	**0**	**58**	**30**	**17**			
	Smal			**77**					
	capaciteit smal	0					0		
	capaciteit breed	70	0	58	31	17	176		

Figure E.5: Vehicle allocation from the optimization model for alternative "Without Remise Zuid"

E.2 Statistical Test of the Results from the Model

Independent sample t-test is used here to test the difference between the mean of two independs sample sets. The numbers of replications are limited in this part due to time constraints. Figure E.6 and E.7 show the result of the test. We can see that from "Levene's Test for Equality of Variances", the variances of the average delay time is not significantly different from each other, but the variances of the deadhead-kilometer are significantly different from each other. Future study with larger sample size can be used to verify these assumptions further. These two assumptions should be verified with larger sample size in future study. Due to the fact that the differences between the means are much bigger than the standard deviations of the samples, we can conclude that there is enough confidence that the means of the two groups are significantly different from each other on both dimensions.

T-Test

[DataSet0] C:\v_dropbox\Dropbox\Thesis\result.sav

Group Statistics

	Alternative	N	Mean	Std. Deviation	Std. Error Mean
Deadhead_kilometer	Current	6	1.0450E6	31910.65932	13027.47212
	Remise Z	6	1.2247E6	37113.72552	15151.61499

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									90% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Deadhead_kilometer	Equal variances assumed	.015	.904	-8.994	10	.000	-1.79720E5	19982.15371	-2.15937E5	-1.43503E5
	Equal variances not assumed			-8.994	9.780	.000	-1.79720E5	19982.15371	-2.16020E5	-1.43421E5

Figure E.6: Independent sample t-test: deadhead kilometer

T-Test

```
[DataSet0] C:\v_dropbox\Dropbox\Thesis\result.sav
```

Group Statistics

	Alternative	N	Mean	Std. Deviation	Std. Error Mean
Average_delay_time	Current	6	56.8098	8.30776	3.39163
	Remise Z	6	46.2898	2.29237	.93585

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									90% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Average_delay_time	Equal variances assumed	7.930	.018	2.990	10	.014	10.51993	3.51837	4.14301	16.89685
	Equal variances not assumed			2.990	5.757	.026	10.51993	3.51837	3.63106	17.40880

```
T-TEST GROUPS=Alternative('Current' 'Remise Z')
  /MISSING=ANALYSIS
  /VARIABLES=Deadhead_kilometer
  /CRITERIA=CI(.9000).
```

Figure E.7: Independent sample t-test: average delay time

F Examples of Deadhead Routes

Line 1:

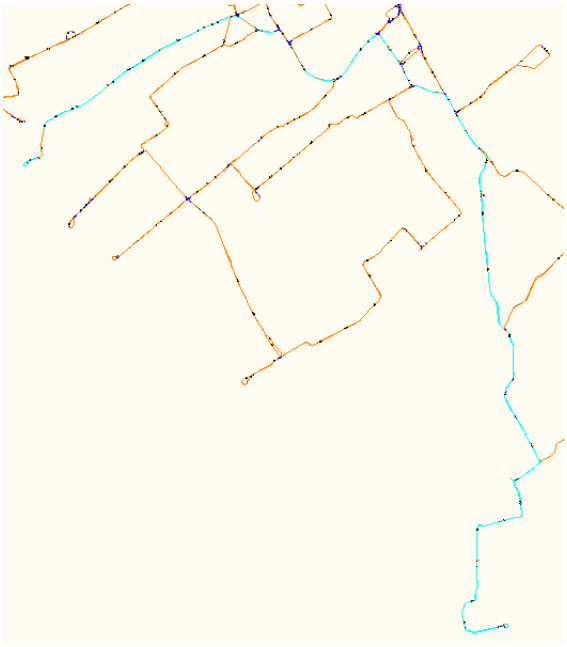


Figure F.1: DeadheadZichtenburgAbtswoudsepark

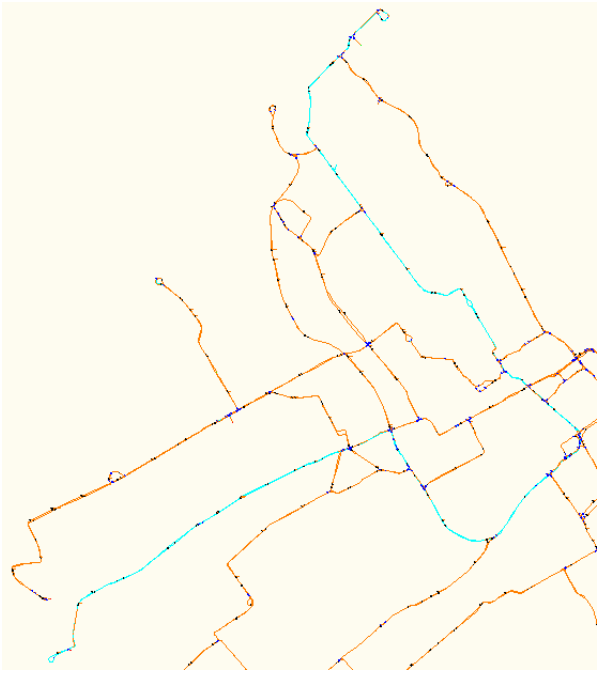


Figure F.2: DeadheadZichtenburgZwartePad