MULTI AGENT BASED TRAIN SIMULATION

JISKE VAN DER LOF

Artificial Intelligence
Information and computing science
Faculty of Science

October 2014 – August 2015

Jiske van der Lof: *Multi agent based train simulation,*
#3407918
October 2014 – August 2015

LOCATION:
Utrecht

Gerard Boersema, EXTERNAL DAILY SUPERVISOR
Medhi Dastani, SUPERVISOR
John-Jules Meyer, SECOND READER

TIME FRAME:
October 2014 – August 2015

ABSTRACT

This research project contributes to the development process of a tool which, based on the current location of the trains in the Dutch railway network, can predict the future locations of the trains in the railway network. This tool is a concept of the Dutch software development company Ordina. For this tool, an agent-based approach is used to simulate the railway scenarios. In order to do so a conceptual model of the railway system is developed. This model is simplified in order to be able to develop an executable model to simulate with. Furthermore it is explained what data should be up to date from the real system in order to be able to make a good prediction. It is explained that at the start of a simulation the values of those parameters which would normally be set by observations of the environment should be set. These values should be set as if the agents were already active in the past. The simplified model is evaluated and recommendations for further research are given.

ACKNOWLEDGEMENTS

I would like to thank my daily supervisor Gerard Boersema at Ordina Smart technologies who gave me the opportunity to work with him on this project and for his help during the various stages of this project. Furthermore, I would like to thank my professor Mehdi Dastani for his support. He helped me to focus on the scientific relevance of this thesis. Also I like to express my gratitude to John-Jules Meyer for his effort as second reader.

This acknowledgment would not be complete without a word of appreciation to my colleagues at Ordina Smart Technologies whom I could always ask for help. Finally, I want to thank my friends and family for their support, with special acknowledgments to Jan, Katja and Willem.

CONTENTS

LIST OF FIGURES

GLOSSARY OF RAILWAY TERMS

The list below translates and explains all railway terms used in my thesis.

ENGLISCH	DUTCH	EXPLANATION
ARI	Automatisch rijweg instelling	Supporting software for local traffic controllers to set routes for trains.
ATB	Automatisch trein beinvloeding	Safety system which makes the train break if a block is entered while violating the speed limit and stops the train if it passes a red signal without permission.
Block	Blok	Area between two signals, in which only one train at a time may be in the area.
Buffer stop	Stootjuk	Construction which block the physical end of a branch.
Control point	Dienstregelpunt	Area which is used in the time table, typically a station or a stop.
Control post	Controllepost	Workplace from which both the local and regional traffic controllers operate.
Delay	Vertraging	Difference between plannened and actual passing time at control point.
Driver	Machinist	The person who operates the train.
Free track	Vrije baan	Consecutive blocks without switches.
Infrastructure	Infrastructuur	The position of the railway.
Joints	Las	The boundary of two sections.
Local traffic controller	Treindienstleider	Traffic controller responsible for one or several PPLG's.
National traffic controller	Landelijke verkeersleider	Traffic controller responsible to maintain good traffic flow and supervise the regional traffic controllers.

(*Continued on next page*)

OCCR	Operationeel Controle Centrum Rail	Workplace of national traffic controllers and coordination point in case of emergencies.
Passage	Passage	If a train passes a control point without stopping at that point.
Planning rule	Planregel	Information about inbound and outbound tracks and time when the route is set by ARI.
PPLG	Primair procesleidings gebied	Collection of neighboring controlled blocks, surrounded by free tracks.This collection is the area a local traffic controller is responsible for.
Railway track	Spoor	All material on which trains can move
Track	Spoor	A part of the railway which can be an origin or destination of a train movement.
Train	Trein	Combination of connected rolling stock which is active on the railway .
Regional traffic controller	Decentrale verkeersleider	Traffic controller responsible to maintain good traffic flow in the area of its control post and supervises its local traffic controllers.
Rolling stock	Rijdend materieel	Smallest part of a physical train which can function as a train on its own.
Route	Rijweg	String decision points on the railway.
Route setting	Rijweginstelling	Action by ARI or local train traffic controller which sets a route for a specific train.
Section	Sectie	Smallest part in which railway is divided.
Slot	Slot	Time interval in which a block is reserved for a train.
Station	Station	Control point where trains can stop, passengers can board and disembark and has at least one switch.

(*Continued on next page*)

Stop	Halte	Control point where trains can stop, passengers can board and disembark, no switch.
Switch	Wissel	Point on railway on which a train can either take the left or right railway track.
Time table	Spoorboekje	Arrival time at each station or stop of each train.
Train activity	Treinbeweging	Movement of train.
Transport companies	Vervoerders	Companies responsible for rolling stock and staff on rolling stock.
Variable signal	Lichtsein	Signal which can show red, yellow and green light.
Yard	PPLG	see PPLG.
Dispatching strategy	versperrings-maatregel	If some lines are (partially) deleted from the time table by OCCR.

1 INTRODUCTION

Train traffic delay and canceled trains are a nuisance to modern society. Delays lead to unsatisfied passengers, economic costs and a bad reputation for ProRail and the transport companies such as NS. Therefore, preventing and solving these delays have a high priority for all parties.

This research project contributes to the development process of a tool which, based on the current location of the trains in the Dutch railway network, can predict the future locations of the trains in the railway network. This tool is a concept of the Dutch software development company Ordina. A prototype of the tool is developed in this research project. For this tool, an agent-based approach is used to simulate the railway system. In the first section of this chapter the scientific relevance, the relevance for the railway system is motivated and the commercial relevance for Ordina is motivated. In the second section of this chapter the research question is formulated and the sub questions are stated that will be answered throughout this thesis. An hypothesis is given of what the answer to the research question will be. Finally, the outline for the rest of this thesis is detailed.

1.1 MOTIVATION

The relevance of the thesis' subject is threefold. Firstly, there is a scientific interest related to agent-based technology. Secondly, the subject is of considerable public interest, since it can partly solve train traffic delay. Thirdly, the subject is of commercial interest for Ordina. The next paragraphs briefly describe these relevancies.

SCIENTIFIC RELEVANCE This project is scientifically relevant for agent-based technology due to the fact that relevant aspects of the real-time state of the railway system form the starting state of the simulation. Therefore, the multi-agent system has to be able to run in two different modes. Firstly, there is the connected mode in which the events of the agents are directly linked to the events in the real world. Secondly, there is the disconnected mode whereby the agents act as independent agents and create their own events. A technique has to be developed to be able to easily switch between those modes and to compare the states of the system in both modes in order to determine whether the prediction is correct. The application domain of rail traffic control gives an excellent example of a situation in which this technique will be of great use.

PROBLEM SPECIFIC MOTIVATION The domain specific problem concerned in this thesis is that of train traffic delay. The Dutch company ProRail is responsible for train traffic infrastructure and management. It aims to have as less delays as possible. New techniques which will decrease the total amount of delays will therefore be of interest for ProRail.

Several sources confirm that most delays are the result of route conflicts[11], [7]. The Dutch railway infrastructure is one of the most dense train networks of Europe. According to the annual report of 2013 of ProRail[13] the net length was 3.061 km and trains drove 155 million km over the network in 2013. A simple solution to solve the problem of traffic delays would be to extend the network by building a bigger infrastructure. Yet, this is a very costly solution. Therefore, ProRail is looking for other solutions such as decoupling railway lines and improving control procedures[12, p.1766].

Conflict: If two trains have to use the same part of the railway at the same time.

This thesis focusses on the solution domain of improving control procedures to detect conflicts. In literature, it is often pointed out that managing railway traffic is a complex task and automatic conflict signalling systems would be of great help for controllers. Kecman[11] explains that the daily control relies mostly on the predetermined rules and the skills of the train traffic controllers. Traffic controllers, consulted in an interview for this thesis, affirmed that they are strictly bound by the rules laid down in the protocols. As Lo, van den Hoogen and Meijer point out in their paper[12, p.1766], due to the fact that people prefer linear processes over complex systems, controllers need tools to be able to make good decisions. As d'Ariano[7] points out, the controllers should even be looking at train performance on a bigger scale. This would enable them to overlook the effects of a train movement on the complete railway network and see conflicts beforehand. If a conflict is detected beforehand, controllers can dispatch in order to minimize the delay resulting from expected conflict. Hence, an assisting tool which predicts the effects of train movements would be of great use of minimizing delays.

As stated, the tool developed in this thesis is based on agent technology. Why agent-based technology is of great use is discussed more deeply in Chapter 2 on literature review.

COMMERCIAL RELEVANCE This research project is commissioned by Ordina. Ordina develops software for companies in different domains. ProRail is one of its clients. For Ordina it is interesting to explore new techniques that solve problems in a more efficient manner. An agent-based approach is a new technique for Ordina. This thesis will give the company insight in the possibilities this technique has to offer.

This research has been conducted at the innovative department of Ordina, called Smart Technologies. This department investigates whether new techniques have potential for Ordina. Especially for this department it is interesting to see and understand the potential added value of an agent-based approach.

1.2 RESEARCH QUESTION

The goal of this software development project of Ordina is to develop a tool which can predict the future locations of trains in the Dutch railway system based on the current locations of trains and other relevant aspects (e.g. velocity of the train). The prediction should be the result of an agent-based simulation of railway scenarios. In order to develop this tool it is first necessary to understand which aspects of the current state of the trains in the Dutch railway network (e.g. location and velocity) should be taken into account and how this data can be used in the agent-based simulation in order to be able to correctly simulate railway scenarios.

This will be the focus of this research project and therefore the research question is as follows: What aspects of the state of the trains in the Dutch railway network and aspects of the network itself are relevant and what technologies are necessary to make an agent-based simulation tool which is able to correctly simulate railway scenarios which predict (based on the current location of the trains, other relevant aspects of the state of the trains and the railway network) the future locations of the trains in the Dutch railway network?

In order to answer this question the following subquestions have been formulated:

SUB QUESTIONS

1. What elements does the infrastructure of the railway system consists of and what rules apply for trains and drivers in the railway system?

2. Which actors of the railway system (i.e. the original system) have to be included in the agent-based simulation to make sure the result of the simulation of the railway scenarios does not significantly deviate from the scenarios in the original system and what are the responsibilities of these actors?

3. What is the simplest agent-based model possible of which a simulation of the railway scenarios would not significantly deviate from the scenarios which occur in the original system?

4. How to represent the infrastructure and the rolling stock of the railway system in the environment of the multi-agent system?

5. How is the disconnected mode of the tool which can predict the future locations of the trains implemented?

Rolling stock: Smallest part of a physical train which can function as a train on its own.

6. How is the transition between the connected mode and the disconnected mode implemented in the tool which can predict the future locations of the trains implemented?
7. How is the connected mode of the tool which can predict the future locations of the trains implemented?

1.3 HYPOTHESIS

The rolling stock should be able to function in both modes. In the connected mode the location of the rolling stock is determined by the location of the actual rolling stock it represents. In the disconnected mode it is controlled by the driver agent. It is assumed that VIEW, a program of ProRail, derives sufficient data in order to represent the actual state of the model in a correct manner. The agents should be initialized at the moment the disconnected mode is started.

VIEW: the program used by transport companies to watch the actual rolling stock movement anywhere in the Netherlands[3]

1.4 OUTLINE

This thesis consists of three parts: Literature, the Multi-Agent System and Discussion.

The first part consists of two chapters. The purpose of this part is to give the background information necessary to understand the multi-agent system developed in this thesis. Chapter 2 gives an overview of problem-related literature. It is interesting and important to understand and take into account solutions with a different approach to this problem and to related problems. It will furthermore show what the benefits of the agent based approach will be. Chapter 3 introduces the protocol used to develop the tool in order to be able to answer all sub questions and the research question. Furthermore it defines all important terms for agent-based modeling.

The second part contains six chapters. Chapter 4 is an analysis of the railway system in the Netherlands. It gives an overview of all the important aspects of the system necessary for this thesis and answers sub question one and two. Chapter 5 answers sub question three. It translates the analysis made in chapter four into a simpler model which is the basis of the executable model which is implemented. Chapter 6 describes the implementation of the environment of the MAS and answers sub question four. Chapter 7 describes the implementation of the disconnected mode including the implementation of the agents. This answers sub question five. Chapter 8 describes what should happen in the transition phase and how this is implemented. This answers sub question six. Chapter 9 describes how the events derived from the actual system about the locations of rolling stock are processed in the connected mode and answers sub question seven.

In the last part, the system as implemented is evaluated. This is followed by the conclusion and recommendation for further research and development of the tool.

In the appendix, the reports of the interviews for this project can be found.

Part I

LITERATURE

2 PROBLEM RELATED LITERATURE

Not much research has been done on the difficult topic of predicting the future location of trains in a railway network. Especially not in the agent technology domain. In order to obtain an adequate overview of existing approaches aimed at solving the issue, related research is examined, which contributes to finding a decent solution for this problem. This chapter is divided in two parts. In the first part, papers are discussed which propose solutions to the problem of predicting aspects of the future state of the railway network (or related problems). The approaches used in these articles are not agent based. In the second part, only papers from the research field of multi-agent systems are discussed. These papers display the utility of an agent-based approach to solve this problem.

2.1 OVERVIEW PROBLEM SPECIFIC RESEARCH

Due to the fact that is hard to physically expand the railway network, one must look for other solutions to be able to transport more people and freight over the existing network. However, most literature is about off-line scheduling instead of dispatching and predicting aspects of the future state. A short description of the most interesting articles with respect to this problem is given below. In the consecutive paragraph the different subproblems and proposed solutions from the articles are discussed. This section concludes with a paragraph in which the shortcomings of these approaches are discussed.

The two most relevant research projects are both Ph.D. theses of the research school TRAIL. TRAIL is the Netherlands Research School on Transport, Infrastructure and Logistics [2]. In 2014 Kecman finished his Ph.D. thesis in the TRAIL Thesis Series. The title is: *Models for Predictive Railway Traffic Management*[11]. It proposes a solution to the problem of predicting railway traffic by making use of the available data of the history of the railway system. Kecman developed a data mining tool which can be used to get interesting data from TROTS. By applying statistical analysis on the derived data, future behavior of the railway system can be predicted. A few years earlier, D'Ariano finished his thesis[7] in the TRAIL thesis series. The title is: *Improving real-time train dispatching: models, algorithms and applications*. It proposes a laboratory tool that can support railway traffic controllers in their task to manage the trains through their areas. The tool checks whether the actual state of the network significantly deviates from the time table. If so, it proposes a dispatching strategy which can be

TROTS: tool via which train traffic controllers can obtain information about the current state implemented by the traffic controller. A third interesting approach is proposed by Espinosa-Aranda and García-Ródenas in the article[10]: *A discrete event-based simulation model for real-time traffic management in railways*. In this article an event-based model of the railway traffic is proposed which supports the traffic controllers in the same way the tool of D'Ariano does. The author argues that due to this event-based model, as opposed to a time-based model, the computional costs of a simulation run will be reduced as much as possible.

REAL TIME TRAFFIC MANAGEMENT The kind of traffic management discussed in this thesis and these papers is what is called real-time traffic management. According to d'Ariano real time traffic management consists of the following[7]: "Modifying the time table, minimizing delays between consecutive trains and ensuring the feasibility of the resulting plan of operations". D'Ariano and also Espinosa-Aranda and García-Ródenas[10] and Kecman[11] state that it is very important to manage traffic in real time since simple initial delays, for example caused by a small technical failure, can propagate so called knock-on delays which in turn can lead to major problems affecting the whole network. D'Ariano defines the real time train dispatching problem as follows[7]:

Definition 1 *Given a railway network, a set of train routes and passing/stopping times at each relevant point in the network, and the position and speed of each train being known at time t_0, find a deadlock-free and conflict-free schedule with no unsolved overlaps between the blocking times of subsequent trains, compatible with their initial positions and such that the selected train routes are not blocked, the speed profiles are acceptable, no train appears in the network before its intended entrance time, no train departs from a relevant point before its scheduled departure time, rolling stock constraints and connected train services are respected, and trains arrive at the relevant points with the smallest possible knock-on delay.*

The goal of dispatching is:

Definition 2 *To minimize train delays while satisfying traffic regulation constraint and guaranteeing the compatibility with the actual position of each train.*

In other words, real-time train dispatching has to find solutions for occurring conflicts by modifying the time table of the conflicting trains without violation of constraints.

RE-ACTIVE VS. PRO-ACTIVE REAL-TIME TRAFFIC MANAGEMENT Real-time traffic management is very complex. Therefore, an aiding tool would be of great use to the traffic controllers. All articles discussed above aim to develop such an aiding tool. However, there is a difference between Kecman's approach and the approaches of d'Ariano and Espinosa-Aranda and García-Ródenas. The last two approaches decide whether dispatching is necessary by looking at the current state. To decide which changes have to be made the tool's predicting ability is used to verify whether a change will lead to more conflicts or to a solution. The real-time traffic management performed in those articles is what Kecman describes as reactive management[11].

Kecman however, uses prediction to detect future conflicts and then changes the time table beforehand. This is called pro-active management. It is important to notice that in Kecman's tools modifications to the time table can also be tested[11] before implemented. As proved in the article of Biederbick, pro-activity significantly improves the dispatching results because conflicts are prevented instead of solved. Biederbick's article is called: *on improving the quality of railway dispatching tasks via agent-based simulation*[5]. This article is further discussed in the next section.

OBTAINING CURRENT STATE IN A TOOL It is necessary to be able to observe and evaluate the current state before one can predict future states. D'Ariano does not go into much detail on how the data of the current state is collected. However, the author does state that the data-loading procedure both has to collect static data, such as infrastructure and time tables, as well as dynamic data on the position and speeds of trains[7]. Espinosa-Aranda and García-Ródenas have based their simulation model on a technology developed by Zandi and Tavana called intelligent transportation systems (ITS)[20]. These systems are supportive systems for dispatchers and consists of several modules. The first module is the load data module, in which data is loaded into the master schedule where it is tested whether the data is still consistent with the time table. This module thus not only observes but also signals whether the actual state deviates significantly from the time table. For this thesis, a data loading system is developed which only collects data. In contrast with d'Ariano and Espinosa-Aranda and García-Ródenas, Kecman widely discusses the way in which data is loaded in its simulation model. Kecman uses the data produced by TROTS to describe the actual state. Traffic controllers use these real-time data streams from TROTS to make dispatching decisions, which explains why Kecman states that in the tool all objects (trains, signals and switches) should be updated in real-time in order to be of help to the traffic controllers. Otherwise the future is not predicted from the actual current state. In Chapter 5 the process mining tool of Kecman will be further discussed. The data produced by TROTS is used by VIEW. VIEW is the tool of Prorail used for this thesis to obtain data.

PREDICTING THE FUTURE To get an accurate tool which predicts the future it might seem reasonable to think that the longer the time horizon, the better the result will be since more information is gathered. There is a certain drawback. The longer the time horizon the more knock-on delays will occur[7], while by solving some of the earlier conflicts later conflicts might not even occur. Thus, some of the calculations where unnecessary. Therefore it is important to make a good trade-off between the size of the time horizon of the traffic prediction and the quality of the prediction.

In d'Ariano's thesis the prediction of the future state of the network is calculated by making use of train speed schedules. Standard acceleration and braking tables of the Dutch infrastructure manager ProRail are used for these schedules. The task of the second module of the intelligent transportation system, as discussed in the article of Espinosa-Aranda and García-Ródenas, is to automatically supervise the train traffic. The algorithm is event-based and assumes the dispatching rule *first come first serve*. So, the first train entering a block gets permission to enter. Consequently, a train arriving one second later has to wait, no matter what the consequences resulting from the blocking may be. So in both articles the rules are very strict and not flexible at all. Kecman introduces a different approach which is more flexible than the approached described above. Kecman states that the variability in process times can be explained by isolating the factors which have a high impact on these process times. Therefore it is of great importance to include all these factors in the prediction of the process times. These actors are for example driving style and peak hours[11]. To be able to take all these factors into account, Kecman makes use of train describer messages produced by the system TROTS in the past. To make an accurate prediction it is important to take the interdependencies between trains that share the same infrastructure or have a planned synchronization constraint into account. The model makes use of statistical models, based on the data collected by TROTS, to predict what is most likely to happen in a particular situation.

SHORTCOMINGS IN THE SOLUTIONS The most important shortcoming of the articles of d'Ariano and Espinosa-Aranda and García-Ródenas is also discussed by Kecman. It is the fact that the estimates of the running and dwell times are computed independently from the actual state of the network. Therefore the predictions of the future state are not accurate enough[11]. Kecman's solution to this problem is to use historical train describer data to develop statistical models. This seems like a very good idea. Nevertheless, a drawback of using historical data, is that only situations that are likely to happen can be predicted with a high certainty. For situations which are more rare it is hard to collect enough data to develop a reliable prediction model. An agent-based approach can be a solution to this drawback.

2.2 AGENT BASED APPROACHES

D'Ariano shortly mentions the idea to use an agent-based approach to solve the dispatching problem. Yet, he is not convinced since "their applicability still needs to be verified by more advanced decentralized or distributed systems"[7]. In this section I will discuss some articles that contribute to this kind of research, and therefor are of interest to this thesis. After a short description of the articles used, the consecutive paragraph explains the author's reasons of for using an agent-based approach to model the railway traffic. The specific problems for which a solution is discussed in these articles is not of interest for this thesis and will therefore not be discussed.

Biederbick and Suhl wrote an article called: Improving the quality of railway dispatching tasks via agent-based simulation[5]. In this article, a customer oriented approach is taken. This approach points out that an agent-based simulation, which makes use of both estimated and real passenger information, can give valuable advice on how to improve the quality of the railway dispatching process.

In *A review on agent-based technology for traffic and transportation*[4] of Bazzan and Klügl an overview is given of the latest research in this field. This paper discusses the main achievements in this research field.

Chen and Cheng[6] state in their article, *A review of the application of agent technology in traffic and transportation systems*, that agent technology specifically can strengthen the design and analysis of the problem domain of traffic and transportation, due to the geographically distributed nature and the dynamic behavior of actors within this environment.

ADVANTAGES OF AGENT-BASED APPROACH Biederbick and Suhl point out in their paper that a railway system consists of autonomous mobile actors, like train drivers and controllers, which act within a dynamic infrastructure. Therefore, they argue, the simulation should model all these components individually. An agent-based model is a natural way to do so[5]. Chen[6] makes the same argument. The main motivations for an agent-based approach are summarized by Bazzan and Klügl[4] and specified below. This summary is coherent with the arguments stated in the other articles.

First of all, it is possible to solve the problem in a natural way since the active entities can have a local perspective. It is therefore not necessary to develop a complex central solution for which it is not possible to include all the necessary details and constraints. Hence, an agent based approach is a solution to the drawback Kecman mentions and partly solves. However, his tool fails to solve this drawback. Secondly, agents are perfect to model a heterogeneous system since all agents can be developed with their own architecture, state and behavior. Thirdly, the possibility to describe agents and their behavior on a high level provides a natural level to talk about the agent-based system and interaction between the system and humans. Finally, agents are able to cope with changes in both the environment as well as the organization system.

Besides these reasons Bazzan and Klügl[4] claim that one of the biggest motivations to use multi-agent systems is to be able to develop and analyze models which are derived from social interactions in human societies. This approach might make it possible to solve conflicts in organizational structures. A subset of agent-based modeling is the one in which cooperating agents are used. According to Bazzan and Klügl this is especially interesting in traffic management[4].

2.3 CONCLUSION

This chapter provided an overview of the related research in order to obtain an adequate overview of other approaches to solve the problem to predict the future location of trains in a network. Due to the fact that it is hard to physically expand the railway network, other solutions are proposed to transport more people and freight over the existing network while reducing the delays. Most literature is about off-line scheduling instead of dispatching and predicting the future locations. The research projects which chose such an approach did some interesting foundings which are of interest for this thesis. Firstly, because it explains that real time train dispatch dispatching is very complex because small decisions can lead to severe consequences. Most of the researches proposed a re-active tool, which signalizes a conflict and tries to solve it afterwards. However, Kecman proposed a tool which is pro-active. This tool does not solve problems but rather prevents problems by predicting the future. This is a good approach, which was also proven by Biederbick.

Furthermore it was explained that the used data should be very accurate. Some of the tools discussed here do not only collect data, but also decide whether or not dispatching is necessary. The latter is not of interest for this thesis.

Yet, it is important to take notice of two details. Firstly, it is important to have a good time horizon for the prediction. The longer the time horizon the more knock-on delays will occur, while by solving some of the earlier conflicts later conflicts might not even occur. Secondly, it is also important to develop a tool in which it is possible to implement rules which are not strict.

The most important drawback of the proposed solutions is the kind of data used to predict the future locations. Most tools compute the estimated running and dwell times independently from the actual state of the network. Kecman does use the current values of the aspects taken into account. However, his tool uses historical data to predict the future. This means that only those problems which occur on a regular basis can be predicted. It is preferred to be able to predict any kind of problem. An agent-based approach makes this possible since agents will simulate the behavior of individual entities in the system. This makes it possible to solve the problem in a natural way since the active entities can have a local perspective. Furthermore, agents are perfect to model a heterogeneous system since all agents can be developed with their own architecture, state and behavior. It is possible to talk about the agent-based system on a natural level. Furthermore the approach to use cooperative agents is of special interest to traffic management.

3 DEFINING MULTI-AGENT SYSTEMS

This chapter discusses the important concepts in agent-based modeling and simulation. First, it will be explained what complex systems, models and simulations are in general and how the latter two can be generated. Afterwards, the basic concepts and definitions of a multi-agent system as chosen for and used in this thesis are explained. This is important, since there is no general agreement on definitions on the concepts discussed here.

3.1 AGENT-BASED MODELING AND SIMULATION

In the first section, defined is what the meaning is of the concepts 'complex system', 'modeling' and 'simulation' as used in this thesis. For developing the model described in this thesis a protocol developed by Siegfried[16] is used.

COMPLEX SYSTEM A system is a complex system if it consist of multiple heterogeneous, interacting components. The behavior of the system is not controlled by any of these components alone. The behavior of the system emerges from the behavior of all the individual components together, which might interact with each other. Because of this, it is very difficult to derive the system's overall behavior by looking at the behavior of the individual components[16].

MODEL A model is a representation of some system, for example a complex system. The aim of a good model is to describe the system in a way which simplifies the system without loosing important aspects. It should provide an easier explanation or analysis of the original system [16]. A developed model can be used to do research on the original system in several ways. The system might be analyzed by making use of mathematical techniques or computer simulation. For this thesis, the technique of computer simulation is used. Siegfried explains in his book that a model should not be seen as something static. It should be seen as a process consisting of seven different stages[16]. These stages are discussed below and it is explained and justified if and how these stages are completed in this thesis.

The development process starts with the sponsor needs. The problem as the sponsor or client has it in its mind is explained by the sponsor.

Secondly, a structured problem description will be developed by the sponsor and the model developers together. To be able to make the model it might be necessary to obtain additional information besides the information resulted from the first stage.

After this second stage it should be clear what specific problem has to be solved and under what constraints. In this thesis there is no actual sponsor or client, besides Ordina, which is better seen as a supervisor of this project then an actual sponsor or client. The problem, for which a solution is searched in this thesis, is discussed with ProRail and the outcome of these interviews is incorporated in the problem description which is described in Chapter 1.

In the third stage a conceptual model is developed, which is the result of a system analysis. It contains a description of all the individual components of the system and how they are related to each other. Also the behavior of all the components is discussed. The conceptual model of this thesis can be found in chapter Chapter 4.

After the conceptual model is developed, it should be checked whether it contains too much or too little information. Therefore, this process should be iterated before proceeding to the next step. From here this thesis takes a slightly different approach than is proposed by Siegfried. This will be discussed after the other steps are explained. For this thesis the conceptual model is very much simplified. This simplification is discussed in Chapter 5.

After this simplified model is developed, it has to be translated into an executable model. In this project this is a java based multi-agent system. The implementation should be the same as the model. Yet, due to programming issues there might be some slight differences. The implementation is discussed in Chapter 6 and Chapter 7. Chapter 8 discusses the implementation of the transition to initialize the agents in the right manner. Chapter 9 discusses the connected mode which provides the data in order to be able to start the simulation at any time.

After the model is executed and tested the necessary amount of times, the results should be interpreted and translated into an answer to the question developed by the sponsor.

ADJUSTMENTS OF THE MODEL PROCESS The first and most important adjustment is the manner of iterating. Siegfried[16] proposes to only iterate the fourth step. However, to test whether the simplified model correctly represents the original system the model should be tested. Therefore, it is proposed to iterate after all steps are processed and the model is tested. If the outcomes are not correct an iteration with adjustments to the model in step four should be made until the results of the tests are correct.

The latter adjustment is about the last step. The request of the client is not to get general answers about the system. The reason for this, is that the executable simulation will be used by the user itself, ProRail, to enable them to constantly be able to predict the amount of railway traffic.

SIMULATION A simulation is a special kind of an executable model. Therefore, a simulation can be made of a complex system by generating a model out of it and, consequently, by executing that model. It is important to differentiate between a simulation run and the simulation engine. The simulation engine is the software application that executes the simulation of the model. According to Siegfried[16] making this distinction has the benefit of being able to run the same simulation of a certain model on different simulation engines. Only if these simulation runs have similar results, both engines could be correct. This distinction is interesting since it forces a developer to make a distinction between the formal model and the implemented model. Therefore the model process remains clear.

3.2 MULTI-AGENT-BASED MODEL

The model and process associated with the model as discussed above is very general. This section discusses the specific model domain used in this thesis. A multi-agent model. In this kind of model the original system is represented by a multi-agent system. The benefits according to literature of using an agent-based approach can be found in Chapter 2. In Chapter 10 it will be discussed which other benefits are found.

In the following paragraph the important aspects of a multi-agent system will be discussed in order to make clear what definitions and concepts are used in this thesis. First a general idea of a multi-agent system is given.

A multi-agent system (MAS) is a system in which intelligent agents behave autonomously. An agent usually represents a particular, autonomous item of a certain system[5]. The agents interact with and have effect on each other and the environment in which they occur. Due to this effect both the environment and the agents can be in different states. Different agents might act differently in the same situation and therefore have a different effect on the environment and each other. Figure 1 gives a general representation of what a MAS should look like[19]. Due to the ability to have natural relations between agents in a MAS and the actors in the original system a MAS also enables the possibility to let the original system take part in the simulation[5].

There are two levels on which an agent-based model can be described: the micro- and the macro-level. The micro-level is mostly about the structure and behavior of single agents, while the macro-level is about the integration of agents and objects into the environment and the interaction between all of them.[16]

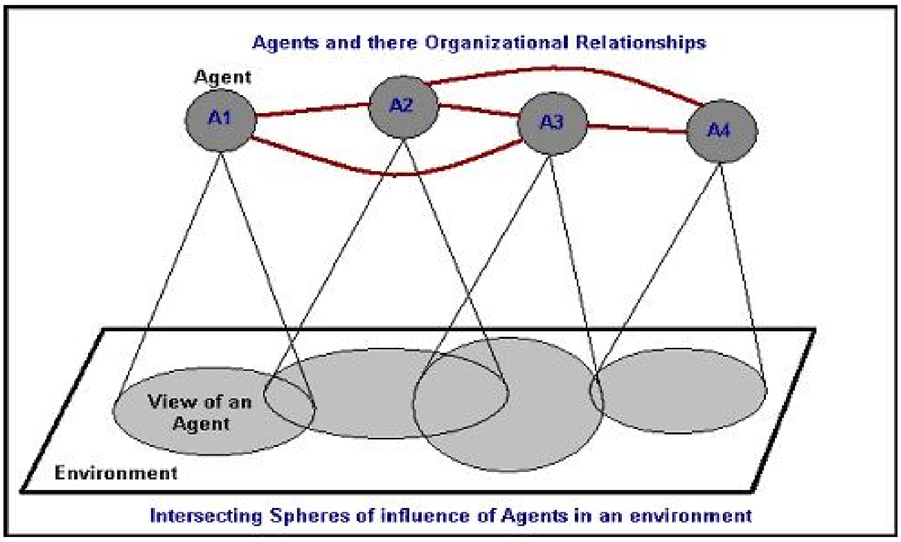

Agents and there Organizational Relationships
Agent
A1
A2
A3
A4
View of an Agent
Environment
Intersecting Spheres of influence of Agents in an environment

Figure 1: Representation of a multi-agent system[1]

AGENTS Many different specific definitions of agents occur in literature. Nevertheless, they mostly overlap. The definition used in this thesis is stated below and orginates from a paper by Woolridge and Jennings which was published in 1995. Woolridge reuses this in its book An Introduction to Multi Agent systems in 2009[19].

Definition 3 *An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.*

According to Siegfried[16] and Woolridge[19], the following characteristics and capabilities can be derived from this definitions. First of all agents are identifiable, discrete individuals. They exist independent of other agents. Secondly, agents are space-aware since they are situated in some kind of environment. Furthermore, the behavior of the agent can be divided into two parts. Interaction with the environment or other agents. And autonomous reasoning about possible actions to perform in order to achieve their objectives[8]. This distinction is similar to the two levels on which an agent-based model can be described, which is discussed above. The interaction is the macro-level and the autonomous reasoning is the micro-level. To interact with the environment, agents will have to be able to perceive the environment in some way and adjust it by certain actions. This will be discussed in the paragraph *Interaction*. The paragraph *Reason* will discuss how agents can decide what action to perform and when.

Learning is left out of this definition even though many other definitions include it. Neither will it be further discussed in this thesis. Learning is not a necessity for an agent. The agents developed in this thesis will not have the ability to learn either.

ENVIRONMENT The environment can best be seen as the background which provides the common playground on which the agents exist [16]. Different types of environments are possible. Spatial environments are most common, however for some problems other types of environments are more useful such as graph-like environments [16]. The environment in the MAS developed for this thesis will also be graph-like.

Besides the difference in type as described above, environments also have different kinds of properties. In the paper Artificial Intelligence: a Modem Approach, of Russell and Norvig 1995 a classification of environment properties for environments is described. Wooldrige uses this classification to define all different possible environments [19].

- Accessible versus inaccessible: This is about obtaining information from the environment. In an accessible environment an agent knows everything about its environment. In a completely inaccessible environment an agent cannot obtain any information from the environment at all. Most environments are partially accessible.

- Deterministic versus non-deterministic: This is about the effects of actions of agents on the environment. In a deterministic environment every action has one guaranteed effect. Therefore there is no uncertainty about the outcome of an action. Neither is there uncertainty about the state the environment will be in after a certain action.

- Static versus dynamic: This is about the ability to change of the environment itself. If the environment is static its state can only change by an action of some agent. If it is dynamic it can change without actions of agents.

- Discrete versus continuous: An environment is only discrete if the number of possible actions and percepts in it are a fixed and finite.

Objects are a special component of the environment. The term object refers to entities within environments that are identifiable and discrete. However, in opposition to agents they cannot behave autonomously. Yet, they can be altered by agents. It is up to the model developer to decide whether something should be part of the environment or to be made an object. A good indicator is the amount of interaction between an agent and the entity. If the interaction is high, it is often constructive to model these entities as objects. [16]

INTERACTION WITH ENVIRONMENT Agents interact with the environment by perceiving it and by changing the state of the environment by performing actions on it. Below these two ways of interactions are discussed further. Figure 2 graphically shows this interaction.

The first kind of interaction is perception. This is very important since it is the only way for agents to obtain information about the state of the environment. This means agents explicitly have to perceive information about the environment. This information can either be actively or passively obtained by the agent by using its sensors. In the first case the agent asks for the information and in the second case the information is pushed to the agent. It is important to notice that the information an agent receives, be it either passively or actively, can be erroneous. Whether this is possible or not depends on the model purpose and the sensors by which the agent perceives [16]. This means an agent has an internal representation of the environment it is situated in. This is part of its believe-base. Its believe-base will be discussed further in paragraph *Reason about actions*.

The second way of interaction is action. Agents have a set of possible actions available to them. This is called an agent's effectoric capability. Not every action can be performed in any state of the environment. Therefore some actions have preconditions associated with them. These preconditions define the necessities to which the state of the environment should hold to perform a certain action. These are also called constraints. Constraints or preconditions can appear in two ways. They can either be limiting conditions as well as assertions for specific properties[16]. It is important to notice that even though the preconditions hold to perform a certain action, the environment might be in another state than expected by the agent after performing that action. This means the agent has, in most environments, only partial control over its environment. [19, p.32]

In a simulation run these two ways of interacting will follow up each other. An agent is triggered by a certain state of the environment and therefore a certain action is triggered. Which leads to another state, which might trigger the agent to update its internal representation of the environment again. Siegfried[16] calls this the sensor-effector chain . Thus, what basically happens is that an agent senses its environment, reasons about it and then makes a plan on how to act within and react on the environment[8]. What is important for these plans is discussed now.

REASON ABOUT ACTIONS To decide which actions from its effectoric capability an agent should put in its plan in order to achieve the desired state, an agent should have some sort of internal reasoning system. The agents' internal representation of the system should be viewed as its belief base which it will use to reason and to make intelligent decisions[8]. This thesis does not try to define intelligence. It focuses, just like Wooldridge and Jennings, on the kind of capabilities that might be expected of an intelligent agent. The capabilities expected are re-activity, pro-activeness and social ability. [19] All these capabilities use the agents believe base to reason and decide what to do.

Re-activity represents the ability of an agent to observe its environment or parts of it in a correct manner and act upon it if changes occur. The actions performed should lead the agent closer to its objectives.

The second capability is pro-activeness. This represents the ability of an agent to perform actions without being triggered by something externally in order to reach its goal. The agent should take initiative and check whether it achieved its goal. If it did not it should try to perform actions to do so.

The last capability is the social ability. This represents the ability to interact with other agents. In some systems an agent needs to cooperate or compete with other agents to achieve its design objectives. However, to do so the agent has to have the ability to reason about an other agent its believes. In case all agents are cooperating it make sense to at least design the agents in such a way that they will never lie.

The challenge is to design agents that have the right level of reactive and goal directed behavior.

In order to reason in the correct manner it is important for an agent to take the entire history of the system into account. A certain decision might seem reasonable at the present, however looking at a broader perspective the decision or behavior might seem unreasonable[19, p.22]. This interaction between agents and how to control the overall behavior of the system is discussed in the next paragraph.

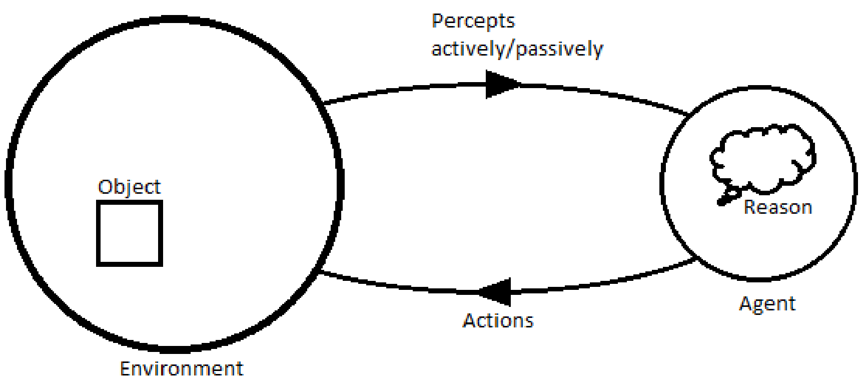


Figure 2: Abstract overview of agent

Figure 2 is an abstract overview of the definition an agent and the interaction with the environment. Objects are included as well.

multi agent level As shortly mentioned above there are multiple possible ways of interaction between agents[19]. It is important to understand what kind of interaction is going on between the system's agents since the internal reasoning of the agents will depend on it. Sichman et al., who is cited by Wooldridge[19] describes four different types of interaction and defines a dependence relation as follows: "The basic idea is that a dependence relation exists between two agent if at least one of the agents requires the other to achieve one of it goals". The four types are:

- Independence. Both agents do not need the other agent to achieve any goal.
- Unilateral. Only one agent depends on the other but not the other way around.
- Mutual. The two agents need each other for the same goal.
- Reciprocal dependence. Both agents need each other to achieve some goal. However this goal is not necessarily the same goal.

Another way to look at the dependency relation is to see whether both contributing agents believe these relations exist. It is called a mutual belief if both agents believe the dependency exist. If only one of the agents believes it is called a local belief.[19]

Another important issue to take into account while developing a multi-agent system is to define the role of an agent in the system. It should be defined what the functionalities, activities and responsibilities of an particular agent are.

Besides the objectives of the individual agents, the system as a whole might have some objectives as well. Instead of only using communication between agents to achieve these objectives an organization which coordinates the whole system in some way might be interesting[4]. The organization could be used to overlook the objectives of the whole system and make sure these are achieved[8].

3.3 CONCLUSION

This chapter discussed and explained the most important concepts in agent-based modeling and simulation. A complex system is defined as a system which consists of multiple heterogeneous, interacting components. A model is defined as a representation of a system. The model process of Siegfried and its six steps are explained. These steps are: defining the clients needs, develop a structured problem description, develop a conceptual model, develop the formal model, translation to executable model, answer the problem. A simulation is defined as a special kind of executable model. The approach of Siegfried is slightly adjusted. Step four until the last step should be iterated until a model is developed that is a simplified version of the conceptual model. The behavior of the executable model of that simplified version should be sufficiently equivalent to the original system it represents.

Furthermore, a multi-agent-based model was discussed in more detail. A multi-agent system is defined as a system in which intelligent agents behave autonomously. An agent usually represents a particular, autonomous item of a certain system. The environment is defined as the background which provides the common playground on which the agents exist. The classification of the environment by Wooldridge is discusssed. Furthermore, objects are introduced which represent entities in the environment which are discrete but cannot behave autonomously. Furthermore the interaction between agents and the environment is discussed and the multi agent level perspective is discussed.

Part II

MULTI AGENT SYSTEM

This part of the thesis will discuss step 3, 4 and 5 of the model process of Siegfried. Chapter 4 describes the conceptual model of the system. Chapter 5 describes the simplified model of the first iteration of the model process of Siegfried. Chapter 6 and Chapter 7 respectively describe the environment and the agents of the MAS. The implementation of the MAS is the executable model derived from the simplified model.

As already discussed in Chapter 1 the scientific relevance of this research project is the fact that the simulation should start in the current state of the railway system. Therefore, it is not only necessary to describe only the disconnected mode in which the multi-agent based simulation is running, but also to describe the transition phase and the connected mode. In order to understand what the transition phase should result in, the disconnected mode is discussed first. The implementation of the transition phase is discussed in Chapter 8. By doing so it becomes clear what data should be made available in the connected mode. Therefore, the implementation of the connected mode is discussed last in Chapter 9.

4 ANALYSIS OF THE DUTCH RAILWAY TRAFFIC

This chapter gives an overview of the Dutch railway traffic. This is step three of the model process of Siegfried. This overview will also answer the two first subquestions of this thesis. The first sub question is: what elements does the infrastructure of the railway system consists of and what rules apply for trains and drivers in the railway system? The second sub question is: Which actors of the railway system (i.e. the original system) have to be included in the agent-based simulation to make sure the result of the simulation of the railway scenarios does not significantly deviate from the scenarios in the original system and what are the responsibilities of these actors?

First the infrastructure is described. Thereafter it is described how the train activity is regulated and monitored. Finally, the tasks which have to be fulfilled in order to maintain good traffic flow are described.

Several sources are used in order to obtain this overview. Besides written documentation, employees of ProRail have been interviewed. The information obtained from the different sources often overlaps. Therefore, it is hard to specifically define what information I got from what source. For this reason all sources are introduced and cited below at once.

In 2012 the Dutch parliament commissioned the University of Delft to review the Dutch railway safety. The report[17] entails a detailed explanation of the organization of the Dutch railway. This part of the report was of great use for this part of my thesis. To complete the overview and find translations for Dutch railway terms to English the master thesis of Roeland van Beek[18] and the network statement 2015 of ProRail[14] are used. At ProRail, Van Beek developed a statistical model of traffic control in the Dutch railway. The network statement 2015 is the official document that provides the information required to get access to and be able to use the railway infrastructure. To verify my overview and to clarify some points I interviewed Emiel Sanders[15] and visited the training center of regional traffic controllers[9]. Emiel's Sanders worked at ProRail at the department incident management. See Appendix A for the elaboration of these two meetings.

A graphical representation of the railway system and the responsibilities is shown in Figure 5.

4.1 INFRASTRUCTURE

The infrastructure of the Dutch railway system is a complex system. It can be explained from three different viewpoints. These different viewpoints are used for different purposes. The three viewpoints are safety viewpoint, macro topology and the schematic viewpoint. The following paragraphs will explain the infrastructure in terms of the specific viewpoint and will explain for what purpose it is used. Figure 24 gives an example of an infrastructure in Figure 3a. The other figures show the representation of this infrastructure in the different viewpoints. This infrastructure is used in all examples throughout the rest of this thesis.

SAFETY VIEWPOINT As the name already indicates, the purpose of this viewpoint is safeness. All railway tracks in the Netherlands have sensors. These sensors count the amount of wheels entered and the amount of wheels that exit a specific part of the tracks. These parts, divided by sensors, are called sections. Due to the sensors any section can be marked as either occupied or marked as free in the safety system. The sections are separated by joints. See Figure 3c for the graphic representation of this viewpoint.

Variable signal: A signal along side the railway tracks which can switch lights.

Several adjacent sections together can form a block. A variable signal indicates an entrance point of a block. Since a section can only be in one block, such a signal also indicates an exit point of a block. Only one train at the time is allowed in a block. The variable signals allocate the current speed limit for the train on the forthcoming block.

PPLG: primair procesleidings gebied.

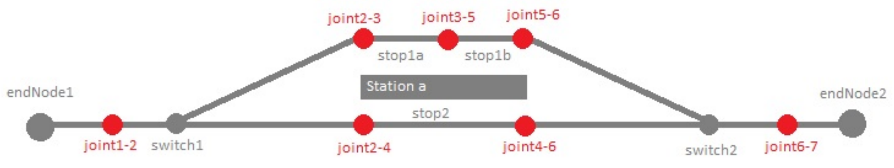
The computer systems used by the local train traffic controllers mostly use this viewpoint. A set of adjacent controlled blocks is called a yard or primary control area, PPLG. Some blocks are controlled automatically. The blocks which are controlled automatically do not contain any switch. A succession of free blocks between two blocks with switches is called a free track.

MACRO TOPOLOGY This viewpoint is mostly used for scheduling and rescheduling. The railway is divided in control points (*dienstregelpunten*) and free tracks. A control point can be a station, a stop, a bridge, etc. Every yard is a control point, however not every control point is a yard. For example, a train station without switches is a free track, however it is also a control point since some trains need to stop there. The macro topology enables an overview without too much detail. It can be used to find optimal and alternative routes. See Figure 3d for the graphic representation of this viewpoint.

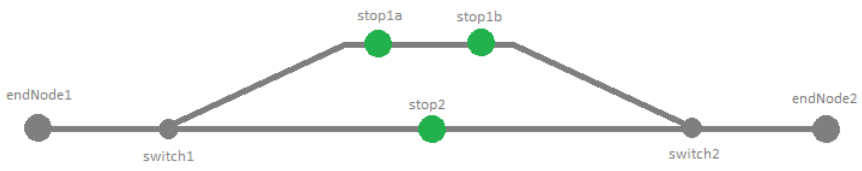
SCHEMATIC VIEWPOINT The schematic viewpoint is actually a more detailed version of the macro topology. The railway infrastructure can be seen as a graph. In the schematic viewpoint a node is a point where the train is able or has to change its direction or behavior. At a switch, a stop and a blind track the train can or has to change its direction. At a joint the train might need to change speed. The edge represents the railway which connects the decision points. See Figure 3b for the graphic representation of this viewpoint.

Blind track: part of railway with a dead end.

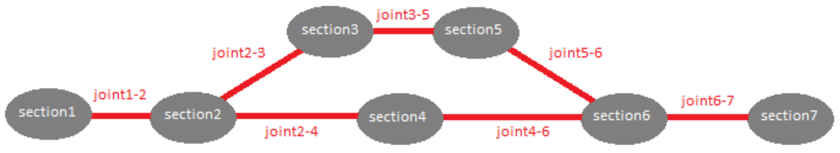
SWITCHES The railway system contains many kind of switches. The simplest switch has the form of a Y. If a train arrives from the tail it can go both directions depending on the position of the switch. It it does not arrive from the tail it will always go to the tail. At other more complex switches there are more options in the position of the switches which result in more options of directions of the train.



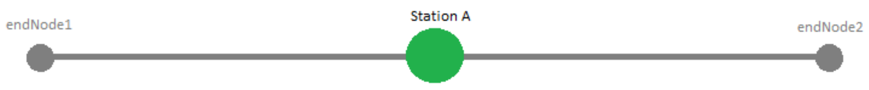
(a) Railway infrastructure



(b) Schematic viewpoint



(c) Safety viewpoint



(d) Macro topology

Figure 3: Viewpoints of railway infrastructure

4.2 TRAIN ACTIVITY

Train activity is regulated by signals and the position of switches. Besides the fact that it is important to know how train activity is regulated, it is also important to understand how train activity is monitored. Train activity can be monitored from two different viewpoints, the microscopic viewpoint and the macroscopic viewpoint. Similarly to the viewpoints in infrastructure, the different viewpoints are used for different purposes. The following two paragraphs explain the train activity in terms of the different viewpoints, for what purposes it is used and by whom it is used. The last paragraph explains how train activity is regulated.

MICROSCOPIC VIEWPOINT The microscopic viewpoint is of great importance for safety. From this viewpoint, a route of a train does not consist of a list of control points. It consists of time-space intervals, called slots. A slot shows on what time interval a certain block, the space interval, is reserved for a specific train. The slots should have some overlap in time and should succeed in space to create a potential route. Figure 4 shows an example of a route from a microscopic viewpoint. Due to the overlap in time and succession in space it is possible to draw a conflict free path in a time-space graph. A slot shows at what time interval a specific block is reserved for a specific train. A safety constraint is ignored if slots that belong to different trains, overlap. After all, only one train is allowed per block. This is called a conflict.

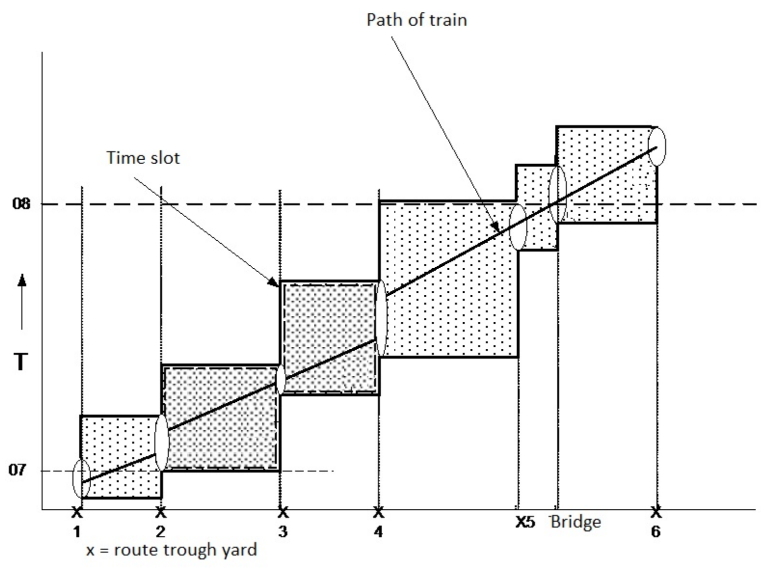


Figure 4: Example route from microscopic viewpoint. This figure is a translated and changed figure from the network statement[14]

MACROSCOPIC VIEWPOINT The macroscopic viewpoint is, just as the macro topology viewpoint of infrastructure as described in the previous section, mostly used for scheduling and rescheduling. It is an overview of all planned rail traffic in terms of departure, passage and arrival times at each control point on the route of the train. The time table only determines the control points the train will pass. The specific sections the train will use is not determined on this level. Normally the time table also shows what track the train should stop at. This viewpoint is used to check whether trains are still on schedule, to develop the schedule and to adjust the schedule if necessary. The exact route of the train is determined by the positions of the switches it crosses. Switches are controlled by local traffic controllers. This is further discussed below.

REGULATION OF TRAIN ACTIVITY There is no steering wheel in a train. Therefore, only the speed of the train can be regulated from within the train. As just stated in the previous paragraph, the guidance is determined by the position of the switches at the railway tracks where the train is on. The driver of the train can regulate the speed, however restrictions to this speed are set by the signals alongside the railway. Two kinds of signals are of relevance in this situation: P-signals and operated signals. P-signals are positioned at the beginning of each block on a free track. Operated signals are positioned at the beginning of each controlled block. In a simplified version signals can show a green, yellow or red light. If the signals shows green the train is allowed to drive the normal maximum speed and the second next block is free as well. If the signal is yellow the train must maintain a speed which enables it to be able to stop before entering the next block, because the next block might be occupied and show a red signal. If the signal shows red the train is not allowed to enter the following block because it is occupied for some reason.

JOURNEY NUMBER Each active train is linked to a journey number. With this journey number it can be identified in all the systems of ProRail. A journey number refers to the scheduled path that train will travel. Several journeys can be executed by the same physical train.

4.3 DIVISION OF RESPONSIBILITIES

Many different parties are involved in the management of railway traffic. As a first division there is ProRail and the transport companies. The transport companies deliver the rolling stock, the staff related to the rolling stock and the stations. ProRail is responsible for the activity of the trains and the railway tracks. The different departments interesting to discuss for this thesis are: driver, local traffic controller, regional traffic controller and national traffic controller. These different departments interact with each other but have their own tasks. There is a certain hierarchy between the different departments. The

Rolling stock: smallest part of a physical train which can function as a train on its own.

following paragraphs will outline what information is available to the different departments, what tasks they perform, what the responsibilities or goals of the different departments are, with whom they can communicate and what their place is in the hierarchy.

DRIVERS Each active train has a driver. He is able to regulate the speed of the train. He knows the planned time table. Adjustments to the planned time table are not automatically made known to the driver. His goal is to follow the time table without violating any constraint. The driver knows the current maximum speed via signals alongside the rail the train drives on. Additional information can be obtained via communication with the local train traffic controller of the area the train is in. Besides a driver is obligated to be acquainted with the topology of the route of the train.

ATB: Autmatisch Trein Beinvloeding

ATB is the system which automatically stops the train if the driver violates a speed limit constraint while entering a block. The driver comes lowest in hierarchy.

LOCAL TRAFFIC CONTROLLERS The local traffic controllers are allocated over 13 control posts throughout the country. Every yard/PPLG is controlled by a local traffic controller. A local traffic controller, or LTC) can have multiple yards to supervise. He is responsible for the route setting of all trains in his assigned PPLG's. His most important task is setting routes for incoming trains while keeping the area safe. He is supported by ARI which is a software program that automatically sets routes within a certain time slot. This timeslot can be manually adjusted if necessary.

ARI: Automatische Rijweg Instelling

The LTC gets information about the current situation via several computer systems. First of all he can access the time table with actual delays. For each train in the area of the controller it gets planning rules which show the arrival time, arrival track and departure track and the time at which the route will be set by ARI. In case of a delay, the controller can decide to modify the rule by changing the route setting time. If necessary, the controller can set a route by hand. However, the safety constraints can never be exceeded. If a later route setting, by ARI or by hand, is in conflict with an earlier route setting, the signals of conflicting blocks will remain red.

The controller has protocols that tell him what to do if a train has a certain delay. However, sometimes he might ignore this protocols and act differently. To make adjustments to the schedule, the LTC typically focuses on the current situation in his own area and on a prediction of the situation in the next 15 minutes. He knows the planned, updated time table.

The LTC can also see what sections of his yards are occupied, what colors the signals show and in what direction the switches are set.

He also sees what trains will have a conflict, namely when they are planned at the same track at the same time.

The local traffic controller communicates with the drivers in its area. Furthermore he will inform the regional traffic controller of his area if he cannot handle the situation on his own. In case of an emergency he will contact the OCCR as well. He has to follow orders from both the regional traffic controller as well as the OCCR.

OCCR: National operational control center

REGIONAL TRAFFIC CONTROLLERS At each control post there is one regional traffic controller (RTC) who is responsible for all areas of the local traffic controllers on the control post. The main task of a regional traffic controller is to make sure delays remain small and to modify the set of available infrastructure in case of a problem. If the RTC gets notified by one of his local train traffic controllers that some part of the infrastructure is no longer available, he has to decide what is going to happen, which trains have to be canceled and what parts of the infrastructure remain available. The RTC makes this changes to the infrastructure in a computer system. This new set of available infrastructure is sent to the schedulers. In the meantime the RTC is responsible to reschedule train paths until the new schedule is sent by the schedulers. The RTC can prioritize trains. The national traffic controller can intervene if the adjustment of one RTC interferes with an adjustment of the set of available infrastructure of another RTC.

NATIONAL TRAFFIC CONTROLLERS The national traffic controller is stationed at the OCCR. The OCCR is responsible to keep flow in the train traffic. Many different involved parties in railway traffic, such as the transport companies and the emergency management department and other departments of ProRail, are stationed at the OCCR together to facilitate collaboration between the different parties if necessary. The NTC constantly overlooks the actions of the regional traffic controllers and helps if necessary. His task is to overlook the management of the regional traffic controllers and to intervene if necessary.

4.4 DELAY

In the Netherlands, the dispatchers acknowledge a train is delayed only when the delayed train is recorded to be at least three minutes behind the schedule[7]; they get no statistical information about the past performance of the same train number, line and timetable reference point, neither about the propagation of delays along the route and lines.

4.5 CONCLUSION

This chapter has given an overview of the Dutch railway traffic. This chapter is the result of step three of the model of Siegfried. Furthermore it answers the first two sub questions of this thesis. The first sub question is: what elements does the infrastructure of the railway system consists of and what rules apply for trains and drivers in the railway system?

The infrastructure has three viewpoints. First there is the schematic viewpoint, in which the topology is represented as a graph in which the edges are railway tracks and the nodes are switches and buffer stops, see Figure 3b. The second viewpoint is the safety viewpoint. This is a graph, in which the edges are the joints and the nodes are the sections Figure 3c. The last viewpoint is the macro topology, see Figure 3d. This is a graph in which the edges are the free tracks and the nodes are the control points. The most important rules which have to be obeyed is that trains can never go faster than the maximum speed of the current section and only one train at the time is allowed in a section.

The second sub question is: Which actors of the railway system (i.e. the original system) have to be included in the agent-based simulation to make sure the result of the simulation of the railway scenarios does not significantly deviate from the scenarios in the original system and what are the responsibilities of these actors?

The driver of the train controls the speed of the train and has a goal to get on time at its scheduled stops without violating any constraints. The local traffic controller is responsible for the safety and has to set the routes of the trains. The regional traffic controller has the task to maintain delays as small as possible. The national traffic controllers are responsible for the general traffic flow. See Figure 5 for a graphical representation of the railway system and the division of responsibilities.

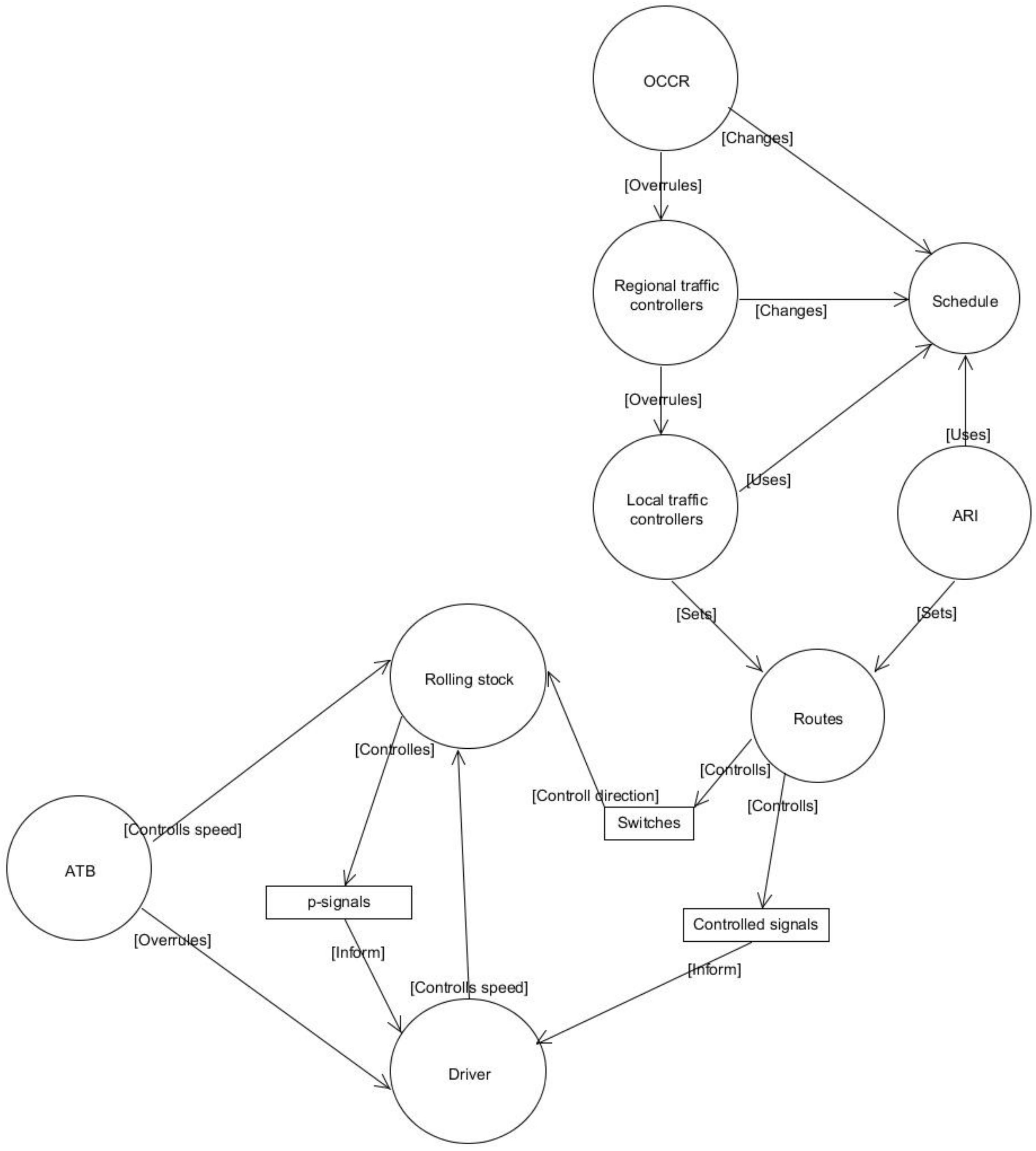


Figure 5: Conceptual model

5 MAS

This chapter will discuss the general design of the program which is developed for this thesis. This chapters contains the result of the evaluation of the conceptual model. This is step four in the model process as described by Siegfried[16]. By executing this step it is possible to formulate the hypothesis of what the answer would be to the following sub question: what is the simplest agent-based model possible of which a simulation of the railway scenarios would not significantly deviate from the scenarios which occur in the original system? In Chapter 10 it is discussed whether this assumption was correct or whether a next iteration is necessary.

As discussed in the introduction of this part of the thesis, the model only describes the simulation of the railway system which occurs in the disconnected mode. The implementation of the transition phase and the connected mode are explained in later chapters.

It is important to keep the agent-based model as simple as possible without loosing any fundamental information. This will keep the model as clear as possible. This is explained in Chapter 3. Therefore this version contains a minimal amount of aspects of the real system. The following sections will explain in what way the railway infrastructure is simplified, how the rolling stock and journey numbers are represented, up until what level of the hierarchy personnel is represented and how other tasks are resolved during a simulation run. This is followed by a concluding section which briefly answers the sub questions.

From now on the term rolling stock will be used to refer to an active train in the railway system.

5.1 DESIGN OF THE PROGRAM

The run of the program which is developed in this thesis starts in the connected mode. In this mode the rolling stock is connected to the live-feed. After the transition phase the program will be in the disconnected mode in which the multi-agent system is active. This is the part of the program in which the simulation will be running. The model of the simulation is discussed below. As explained in Chapter 3 the process of developing, implementing and evaluating a simplified model should be iterated until a model is found that displays similar behavior as the original system it is representing. Figure 6 shows a graphic representation of a run of the program and how the iteration should work. A first simple evaluation of the program is discussed in Chapter 10.

As explained in the introduction to this part of the thesis, the order in which the different modes are discussed are not equivalent to the order in which the modes occur in a program run. It is necessary to understand what the required data for making a transition are, in order to understand what data should be collected in the connected mode. Furthermore, it is necessary to understand what the start state of the disconnected mode should consist of in order to understand what the transition phase should result in. Therefore, first the simplified model of the conceptual model is given below. Thereafter, in the following chapters, the disconnected mode is discussed followed by a chapter about the transition phase. Finally, the connected mode is explained.

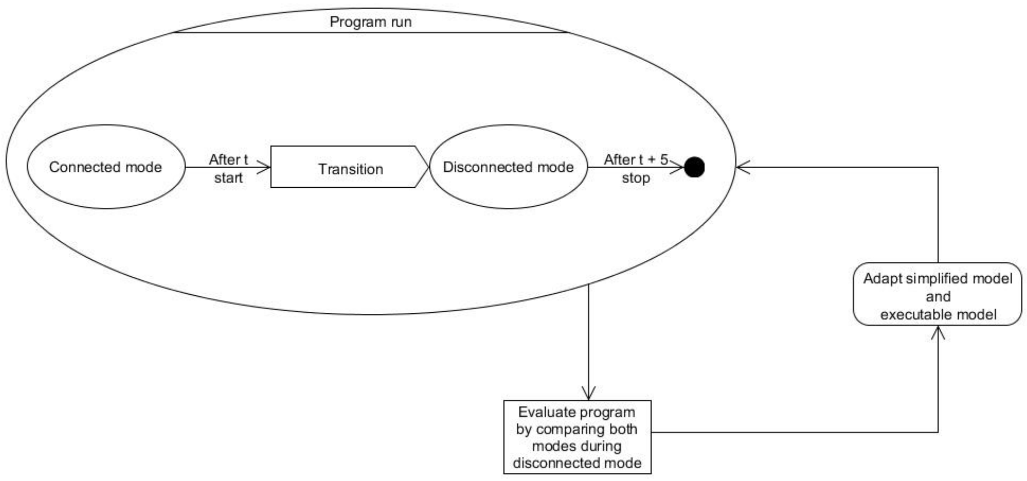

Program run
Connected mode
After t
start
Transition
Disconnected mode
After t + 5
stop
Adapt simplified model and executable model
Evaluate program by comparing both modes during disconnected mode

Figure 6: Run of program and iteration process

5.2 SAFETY CONTROL OF ENVIRONMENT

In the railway system the goal of a signal is to make sure at most one rolling stock is in a block to keep rolling stock at a safe distance from each other. In this model sections do not belong to a block. Blocks do not exist in this model just as signals do not. A section is set occupied once a rolling stock has entered that section. It is automatically set free if the rolling stock leaves the section. All sections which will be passed in the after-running path of a rolling stock are set reserved. This action represents the route setting as is done in the microscopic viewpoint, which is discussed in Chapter 4.

A driver is triggered as soon as a relevant section is blocked by any other rolling stock. This is the representation of signals. The goal of a signal is preserved in this model. This will be further explained in *Driver*. Furthermore, a driver is allowed to let the rolling stock have a maximal velocity with what it will be able to change its velocity in order not to violate any constraint in the future.

5.3 RAILWAY INFRASTRUCTURE

Only a part of the infrastructure is taken into account. The focus is on the yard Amersfoort. It is explained in the next chapter why this decision is made. In Chapter 10 it is discussed whether this is a problem. As discussed in Chapter 4 there are three viewpoints from which the railway can be represented. This program uses two of those representations. The schematic viewpoint is used to represent the actual topology of the railway infrastructure. This is represented as a graph consisting of nodes and edges. This is further discussed below. The second viewpoint is the safety viewpoint. In the program it is known for each part of the railway infrastructure to which section it belongs. This is further discussed in the paragraph *Section*. The macro topology is left out since this only is a less detailed version of the schematic viewpoint. The macro topology is mostly used to schedule the time table. In this program the development of time tables is not taken into account, therefore the macro topology is not of interest to this program. This is further explained in the paragraph *Controllers*.

As already stated. Signals are not directly represented in this implementation. However it is known whether a section is occupied or not. So, it is possible to make sure that only one rolling stock at the time is in a certain section. This precisely is the purpose of signals. Further more, each section has a limit. This limit depends on the distance between the nearest station and the location of the section. With a minimum speed of 40km/h and maximum speed of 140km/h. For this thesis it was not possible to collect the real limits of each section. For each section it is also known what drivers are waiting for the section to be free again. The methods for filling and removing from this map are explained in Chapter 7.

Switches are very much simplified. Any kind of switch is just one point with in- and out-edges. It is not taken into account that only certain transitions from an in-edge to an out-edge are possible if these belong to the same switch node. For this implementation this is not necessary since routes are already set, see paragraph *Controllers* below for further explanation. If later implementations include finding new routes this feature would be necessary to implement. This will be discussed in Chapter 10. Furthermore, it is assumed that a switch is always in the correct position for the rolling stock passing the switch.

Locations of switches, joints and endNodes are defined in rijksdriehoek-coordinates . All switches have the exact location, all joints which coincide with a signal have their exact location and each joint at the end or beginning of a platform have their exact location. All endNodes are located on the node most close by the endNode. It is assumed that the unknown joints in between two known points are evenly divided over the distance of the railway track in between those two known points. It should be checked whether this is precise

Rijkdriehoek-coordinates: Dutch representation system of location compatible with GPS-Coordinates.

enough. This will be elaborated in Chapter 10. Furthermore it is assumed that railway tracks are straight lines between known locations of switches and known locations of joints. The radius of a railway track is always very large. Since in this implementation only the yard Amersfoort is taken into account it can be assumed that tracks are not very long. Therefore any curves would not have a big influence on the distance of the railway track. However, if the complete railway section will be implemented this assumption might lead to problems. This will be further discussed in Chapter 10.

5.4 ROLLING STOCK AND JOURNEY NUMBERS

Rolling stock and journey numbers are united. Each rolling stock will represent a specific journey number. The provision of rolling stock is assumed to be infinite anywhere on the infrastructure.

Apart from this unity, the rolling stock in this model is similarly represented to that in the conceptual model. The speed of the rolling stock is determined by its driver, taking the physical laws into account. The route is determined by the route as set. The rolling stock contains a display on which its state is made visible to its driver. This display contains its current speed and exact location given in section, edge and location on edge. Furthermore, it contains its exact route left and the plan rules. A rolling stock can hold a driver. The rolling stock can be appointed to since it is a discrete entity in the environment. It however, does not have any autonomous behavior.

The length of a rolling stock is not taken into account. It is assumed that the length of a rolling stock does not have much influence on the occupation of sections. The maximum possible speed is assumed to be 200 km/h for any rolling stock. A maximum speed is necessary to make sure rolling stock does not go any faster then possible. The velocity chosen of 200 km/h is higher than the maximum possible speed allowed in any section in this focus. It is assumed all rolling stock is able to achieve a velocity of 200km/h.

5.5 DRIVERS

The controllers of the rolling stock are drivers. Therefore, they are identifiable and discrete individuals. In this paragraph it is discussed which goals are represented in the model, what knowledge the drivers are assumed to have and what its functionalities are.

First of all, the goals of the drivers in the real system are twofold. The first goal is to arrive on the scheduled time at each station it has to stop. Furthermore, its goal is to not violate any safety constraints. In this implementation the first goal is not taken into account since the time table of planned stops is not taken into account. This is explained in the paragraph *Controllers*. However, the second goal is taken into account and translated into the following goal for the driver: a driver will always retain a goal speed which makes sure the current speed is as high as possible without violating any constraints. By doing so, the necessary brake distance is taken into account in order not to violate speed limits on the forth coming route.

The knowledge a driver is assumed to have is slightly different from the real situation. So, first a small recap of the knowledge-base of the driver in the real situation is given. Thereafter, the knowledge-base in this model is explained. In the real situation a driver is aware of the planned schedule on a macro level and is familiar with the railway tracks it will cross. Furthermore, it is able to observe the signals and can communicate with the local traffic controller of the area it is currently in to retain more information if necessary. Furthermore, it can check the speed of the rolling stock and is able to brake or accelerate. The direction of the switches decides which side of the switch is taken by the rolling stock.

In this implementation the location of the rolling stock, exact route, the plan rules and the speed can be looked up by the agent in the rolling stock on the display as described earlier. The agent receives other information about the environment via triggers. A trigger is passed to an agent if the MAS is in a certain state. The trigger activates the process to perform a certain action. Below the different triggers which can occur in this MAS are discussed and their representation is explained. Any time a rolling stock passes a joint its agent will be triggered to adapt the speed of the rolling stock if necessary. The trigger contains information of the current limit, the next limit and the location where the next limit will hold. This represents the observation of a signal, or another speed limit by a driver in the original system. Furthermore, a driver is triggered to let the rolling stock stop if the section right after the stop section of this rolling stock is blocked and the distance between the entrance of the blocked section and the stop location is less than 10 meters. This represents the observation of a red signal in the original system. As soon as a section for which it is waiting is set free it gets a trigger to start driving again. The latter two triggers represent the perception of signals. This represents the observation of a signal turning green in the original system. The action a driver agent can perform is to set the goal speed of the rolling stock. It is assumed that a rolling stock has a parameter which can be set. The deceleration and acceleration of the rolling stock is automatically handled by the rolling stock.

5.6 CONTROLLERS

The most important task of controllers is to set routes and dispatch if necessary. In this implementation none of the controllers are implemented. It is important to see what this relatively simple version of the program can effectuate before making it any more complex. Therefore, in this version all routes are set and cannot be dispatched in any way. The routes are set by looking at the plans pushed by the data streams to the program. It is assumed that a switch is automatically set in the right position if a rolling stock crosses it. The goal of this program is to help traffic controllers prevent conflicts by signalling conflict in the future. Therefore, this prediction of the future does not need to contain dispatching actions since this is outside of the scope of the thesis.

5.7 CONSTRAINTS

The drivers are assumed to be completely honest. Therefore, they would never violate a constraint. For this reason it is unnecessary to implement anything like an organization in this model for it overlooks the agent's behavior. However, this might be an interesting addition to the complexity of the program, especially if the goal of a driver to arrive on time at stops is taken into account. This will further be discussed in Chapter 10.

5.8 REPRESENTATION IN MAS

The multi-agent system exists of an environment containing the railway infrastructure and the rolling stock driving on it. The drivers are the only agents present in this MAS. The following two chapters will elaborate on the reasons for these representations. Figure 7

5.9 CONCLUSION

This chapter provided the model for what the MAS is developed. It explained which parts of the conceptual model are represented in this model. The railway infrastructure is represented as a graph in which the schematic viewpoint and the safety viewpoint are integrated. Signals are not represented in the model, however their goal to allow at most one section in a block to keep rolling stock at a safe distance from each other is preserved. The rolling stock and journey numbers are united. The rolling stock is represented in quite the same way as in the real situation. Furthermore, it contains a display with its current speed, exact location on the graph, to travel route and plan rules. The length of a rolling stock is not taken into account and its maximum speed is set to 200 km/h. A driver has only one goal. This is to never violate a speed constraint. The knowledge of the driver contains of its own state, the limit of the current section, the limit of the next section and it can check the display of the rolling stock any time. Controllers are not represented in this model. Therefore the routes are fixed. This is not a problem for the purpose of this tool, since this is to support the local traffic controllers in their decisions.

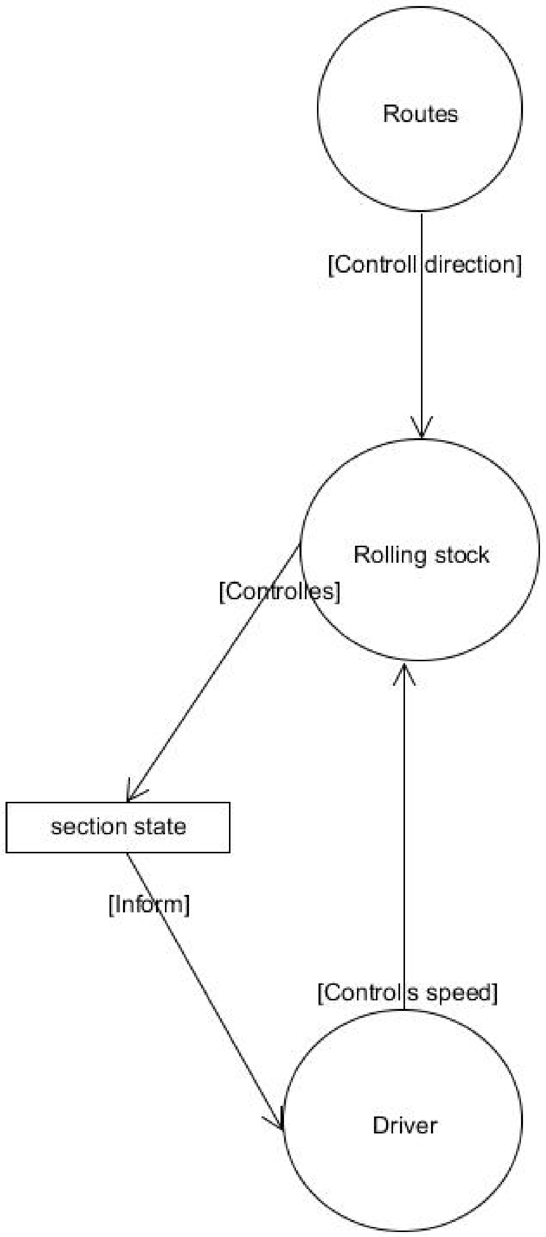


Figure 7: Flowchart of simplified model

6 ENVIRONMENT

As explained in the previous chapter, the environment consists of the railway infrastructure and the rolling stock driving on it. The railway infrastructure is represented as a graph, in which both the safety viewpoint and the schematic viewpoint are represented. Figure 10 shows the UML of the classes in which the environment is implemented.

The sub question answered in this chapter is: how to represent the infrastructure and the rolling stock of the railway system in the environment of the multi-agent system?

The first section explains the environment in terms of the properties as stated by Russel and Norvig and adapted by Wooldridge[19]. This gives a theoretical description of the environment. The second section will discuss how the railway infrastructure is represented and implemented. The third section will elaborate on how the rolling stock is represented and why this is a part of the environment. The fourth section discusses how the state of the environment is controlled and what the rules of the environment are. Thereafter, in section five it is explained how data necessary to create the environment is acquired. This is followed by a concluding section which briefly answers the sub question.

6.1 THEORETICAL DESCRIPTION OF ENVIRONMENT

The environment representing the railway infrastructure and the rolling stock is defined in terms of the properties stated by Russel and Norvig and adapted by Wooldridge[19] and adapted by Wooldridge. The definition of these properties can be found in Chapter 3.

The environment is partly accessible for the agent. The rolling stock, which the driver agent is controlling is completely accessible to the agent. Therefore, the location of the rolling stock, the speed of the rolling stock, the route and the plan rules are accessible to the driver agent. Furthermore, the driver gets information about the limits of the section the rolling stock is in via triggers. Besides, the driver will be triggered as soon as the relevant section is no longer free. However, a driver does not know anything about the location or speed of any other rolling stock in the environment or the infrastructure outside its own route. Therefore, the environment is not completely accessible to the driver agent.

The environment is deterministic. The outcome of an action is predictable if the current state of the complete environment is known.

The environment is dynamic. If a rolling stock has a certain speed no action of the agent is necessary to move the rolling stock to any other location.

The environment is continuous. The exact location of a rolling stock can be anywhere on any edge in the environment.

6.2 RAILWAY INFRASTRUCTURE

As discussed in Chapter 5, only two viewpoints of the infrastructure are represented in this MAS: the schematic viewpoint and the safety viewpoint. These viewpoints are compatible and therefore represented in one graph. Joints, switches and blind stops are represented as nodes. The railway in between those points are represented as edges. The yard Amersfoort contains 98 switches, 263 joints and 215 sections. Furthermore 24 plan rules are known.

NODE A node represents a switch, buffer stop (called endNode) or joint.

Any kind of switch described in Chapter 4 is represented by one node. A switch is either connected to three or four other nodes, depending on the kind of switch it represents. An endNode is connected to exactly one other node. A joint is connected to exaclty two other nodes.

Every node has a name which starts with the kind of node, either 'switch', 'joint' or 'endNode' followed by a structured code depending on the kind of node. See Figure 9 for some examples.

EDGE An edge represents one direction of the railway track between two nodes. Thus, each railway track between two nodes is represented by two edges. It is assumed that the railway track is a straight line between those two nodes, which is explained in the previous chapter. Figure 8 shows the representation of the railway system in the graph.

The name of an edge starts with 'railway' followed by the names of the two nodes it connects and finishes with +0 or +1 to distinguish the directions.

The length is the shortest distance between the two nodes it is connecting.

For each edge it is known from which node it originates and which node is its destination.

For each edge it is known to which section it belongs. The edges are directed to be able to determine the direction of a rolling stock.

SECTION A section is not directly represented in the graph. However, as just stated, for each edge it is known to which section it belongs and joints are represented as nodes in the graph. Therefore, if

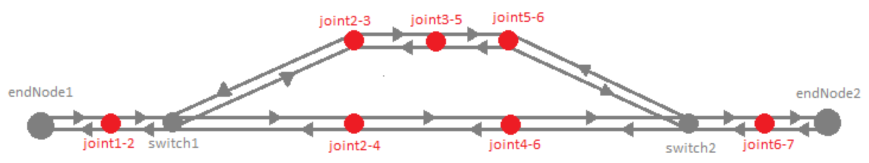


Figure 8: Graph as implemented

it is known at which edge a rolling stock is located, it is known at which section it is located.

The name of a Section starts with 'section' followed by the code of the PPLG and the code of the section as in VIEW.

The class Section is a slightly more complex class with several methods, in order to be able to set the variables *occupiedBy* and *reservedFor*, see Figure 10. The argumentation for the logic of these methods is explained in the following chapters.

6.3 ROLLING STOCK

As already mentioned in the previous chapter, the rolling stock combinations are represented as objects as described in Chapter 3. The rolling stocks are represented as entities which are discrete and identifiable in the environment but cannot behave autonomously. The movement of each rolling stock is handled individually by that object-oriented object of the class Rolling Stock. The methods responsible for movement are discussed in Chapter 7.

6.4 DATA ACQUIRE PROCESS

Acquiring the data necessary to implement the infrastructure of the railway traffic in the program was not planned to be in the scope of this thesis. This nevertheless had to be done, since the necessary data was not available. The data supplied was a graphic map of the infrastructure and events from VIEW, which show the routes traveled by rolling stocks. This had to be manually translated into data from which a graph could be generated by a computer. Furthermore, it was not documented on this map what the names of the sections on this map where and what the exact location of the joints are. This had to be derived from the routes from VIEW. The known locations of switches and some joints were given as implementable data. The others are derived by the program as discussed in the previous chapter. Due to the amount of manual labor and the lack of time it was decided to only implement PPLG Amersfoort in this program. See Figure 11 for the representation of the graph in the GUI of the program. It was assumed that this has not much drawbacks for this thesis. This is further discussed in Chapter 10.

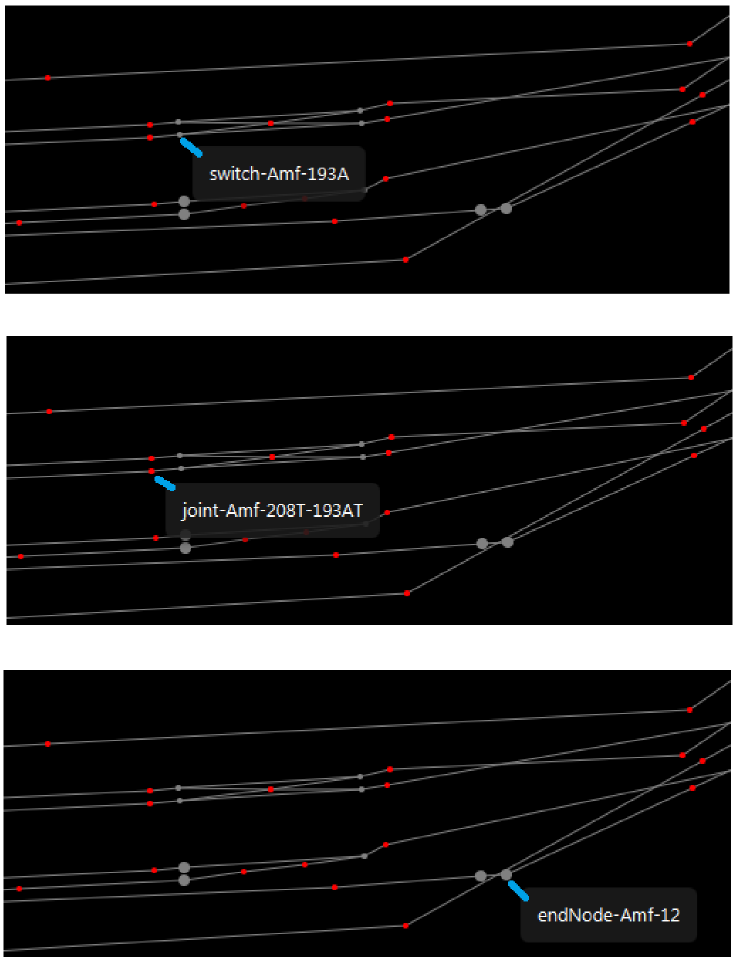


Figure 9: Part of implemented graph

6.5 CONCLUSION

The environment can be described by the properties of Russel and Norvig and adapted by Wooldridge[19] in the following way. It is partly accessible, non deterministic, dynamic and continuous. Some of these properties are doubtful. The sub question answered in this chapter is: how to represent the infrastructure and the rolling stock of the railway system in the environment of the multi-agent system? The railway infrastructure is represented as a directed graph. The edges represent the railway. For each railway track there are two edges one for each direction. The nodes represent the switches, endNodes and joints. A section is not directly present in the graph. However, for each edge it is known to which section it belongs. The rolling stock is represented as an object in the environment since it is an identifiable discrete entity in the environment without autonomous behavior.

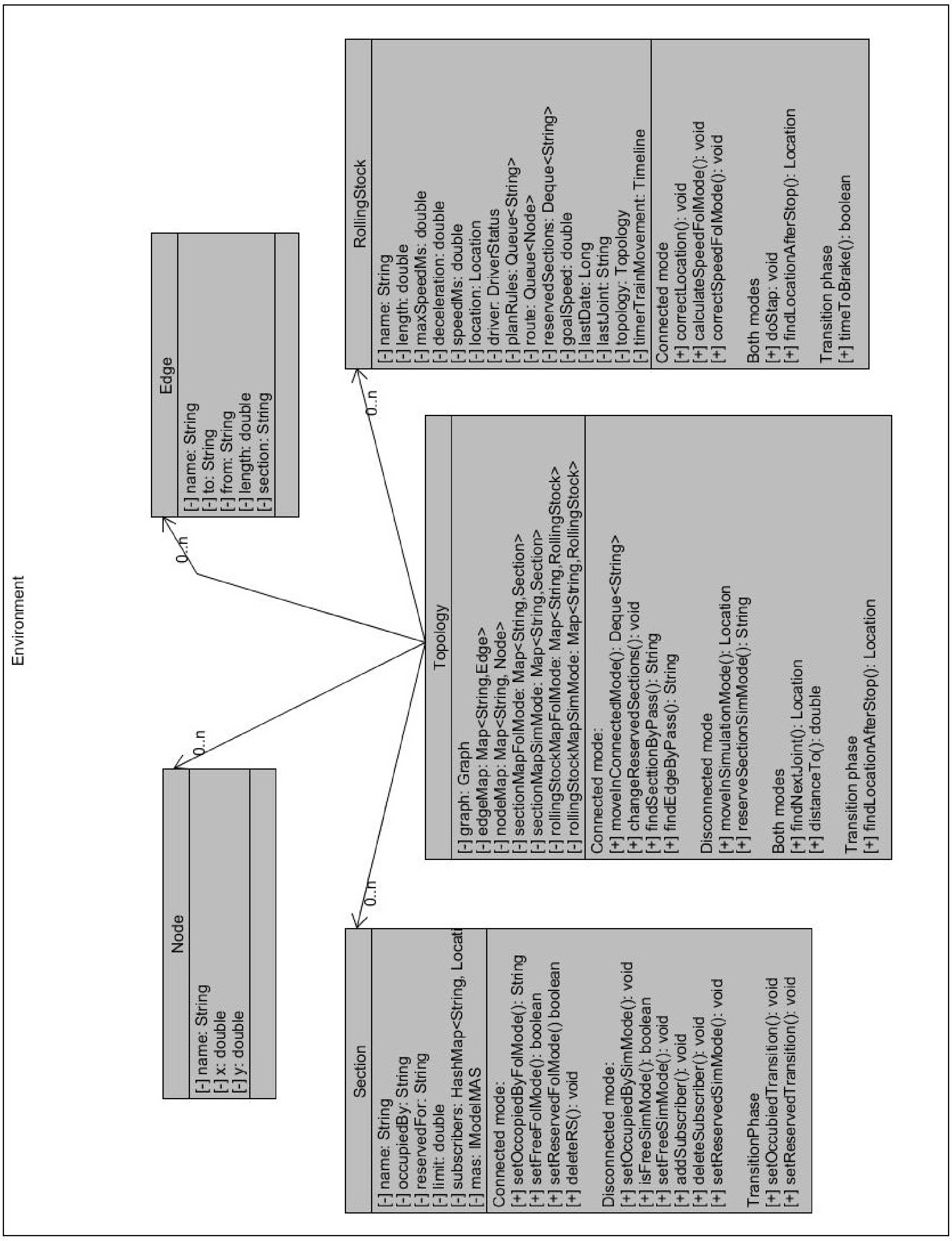

Environment
Edge
[-] name: String
[-] to: String
[-] from: String
[-] length: double
[-] section: String
RollingStock
[-] name: String
[-] length: double
[-] maxSpeedMs: double
[-] deceleration: double
[-] speedMs: double
[-] location: Location
[-] driver: DriverStatus
[-] planRules: Queue<String>
[-] route: Queue<Node>
[-] reservedSections: Deque<String>
[-] goalSpeed: double
[-] lastDate: Long
[-] lastJoint: String
[-] topology: Topology
[-] timerTrainMovement: Timeline
Connected mode
[+] correctLocation(): void
[+] calculateSpeedFolMode(): void
[+] correctSpeedFolMode(): void
Both modes
[+] doStap: void
[+] findLocationAfterStop(): Location
Transition phase
[+] timeToBrake(): boolean
Topology
[-] graph: Graph
[-] edgeMap: Map<String,Edge>
[-] nodeMap: Map<String, Node>
[-] sectionMapFolMode: Map<String,Section>
[-] sectionMapSimMode: Map<String,Section>
[-] rollingStockMapFolMode: Map<String,RollingStock>
[-] rollingStockMapSimMode: Map<String,RollingStock>
Connected mode:
[+] moveInConnectedMode(): Deque<String>
[+] changeReservedSections(): void
[+] findSectionByPass(): String
[+] findEdgeByPass(): String
Disconnected mode
[+] moveInSimulationMode(): Location
[+] reserveSectionSimMode(): String
Both modes
[+] findNextJoint(): Location
[+] distanceTo(): double
Transition phase
[+] findLocationAfterStop(): Location
Node
[-] name: String
[-] x: double
[-] y: double
Section
[-] name: String
[-] occupiedBy: String
[-] reservedFor: String
[-] limit: double
[-] subscribers: HashMap<String, Locati
[-] mas: IModelMAS
Connected mode:
[+] setOccopiedByFolMode(): String
[+] setFreeFolMode(): boolean
[+] setReservedFolMode() boolean
[+] deleteRS(): void
Disconnected mode:
[+] setOccupiedBySimMode(): void
[+] isFreeSimMode(): boolean
[+] setFreeSimMode(): void
[+] addSubscriber(): void
[+] deleteSubscriber(): void
[+] setReservedSimMode(): void
TransitionPhase
[+] setOccubiedTransition(): void
[+] setReservedTransition(): void
0..n
0..n
0..n
0..n

Figure 10: UML Environment package

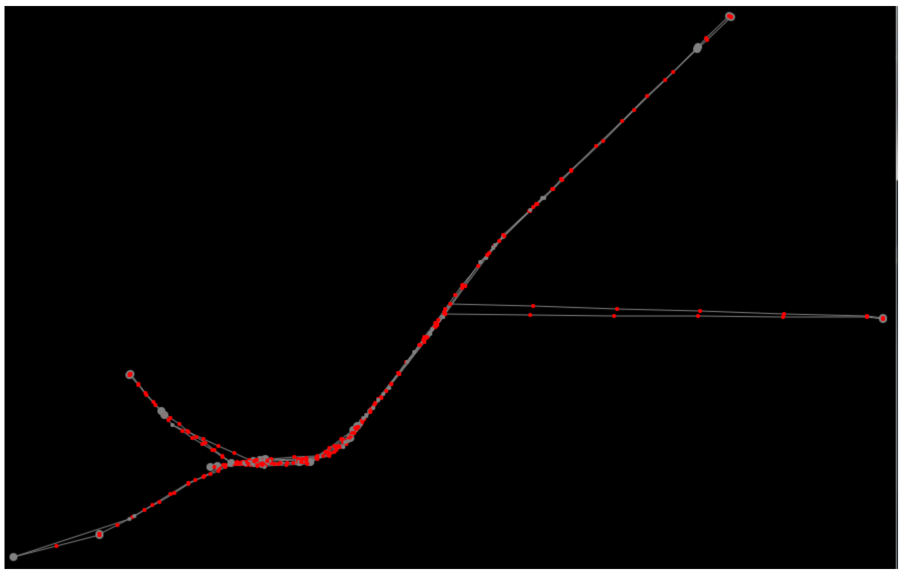


Figure 11: Representation of implemented area in GUI

7 DISCONNECTED MODE

As defined in the introduction to this part of the thesis, the MAS is activated in the disconnected mode. This chapter explains and discusses the architecture and implementation of the agents of this MAS. As explained in Chapter 5 the only kind of agent which occurs in this implementation is the driver agent. The environment of the MAS is discussed in the previous chapter. However, this chapter explains the implementations of the methods in the environment which handle the movement of the rolling stock in the disconnected mode. The sub question that will be answered in this chapter is: how is the disconnected mode of the tool which can predict the future locations of the trains implemented? In order to be able to create the methods of the transition phase, it is shortly discussed in what state the belief base has to be right after the disconnected mode is started.

The first section gives a theoretical description of the driver agent. The second section discusses the design pattern used to develop the agent and discusses the architecture and implementation of the agent. The third section discusses the important methods used throughout the whole program, which handle the actual movement of the rolling stock in the environment. The fourth section explains in what state the belief base has to be right after the disconnected mode is started. The chapter ends with a conclusion.

7.1 THEORETICAL DESCRIPTION OF THE DRIVER AGENT

As described in Chapter 3, its important to understand how the reasoning of agents works, how the interaction with the environment and other agents in the environment works, and what the dependence relation with other agents is like

Concerning the reasoning of the agent, the reactive ability of driver agents is that they observe joint passes and this gives them information about the speed limits relevant at that moment. Furthermore, it has the reactive ability to observe a relevant blocking or release. As discussed in Chapter 5 the driver agent has the goal to always retain a goal speed which makes sure the current speed is as high as possible without violating any constraints. Therefore, its pro-active ability is to constantly check whether its goal speed should be changed. Lastly, in this implementation the driver agent does not have any social abilities. Although the driver agents interact indirectly with other driver agents because of occupied or reserved sections, agents cannot, in any way, communicate directly with other agents.

The interaction with the environment takes place via triggers and executable strategies. The agent can subscribe itself to receive certain triggers which can be seen as the perception of the agent. Furthermore, an agent has some strategies which it can execute if the strategies are relevant for the perceived triggers. The execution of a strategy represents an action. The test whether or not a strategy is applicable represents the preconditions of an action.

Lastly, we look at the multi-agent level. The dependence relation for this implementation, if looked at from the individual goals of the agent, is independent. Agents do not need other agents to maintain their goal of retaining a goal speed which is maximal without violating any speed limit that is currently relevant to the agent. Agents do not need other agents to achieve their goal. An agent can always retain a goal speed which makes sure the current speed is as high as possible without violating any constraints without the help of any other agent. However, the goal of the entire system is to have as few delays as possible. Therefore, it is necessary to work together since it might be possible that giving priority to one rolling stock results in less delays. Therefore, a different rolling stock should not take the place of the rolling stock that should get priority in order to get as less delays as possible. Thus, when this is taken into account, the dependence relation is mutual. Furthermore, this dependency is mutually believed by all agents.

7.2 DRIVER AGENT

The agents are developed by making use of the design patterns for agents of Dastani and Testerink[8]. First, a general outline of the design pattern is given and the most important methods are explained. Second, the implementation of the agent is discussed class by class.

DESIGN PATTERN This object-oriented design pattern is based on the programming construction developed in agent-based programming domain. It is assumed here that beliefs of an agent represent the systems state. The beliefs of the agents are represented in a special class called ContextInformation.

Dastani and Testerink have coined the term *strategy* to describe the term plan. A strategy is triggered by specific triggers if the strategy is relevant to that trigger and that trigger occurs. A strategy is executed if it is applicable; executing a strategy represents an action. The set of executable strategies is thus the effectoric capability of an agent as discussed in Chapter 3. A strategy is applicable if the parameters in the context information are set right. This represents the preconditions of a the strategy. The triggers are processed by methods in the Scheduler class of an agent. The triggers are added to the trigger Queues by the class GoalProxy, which is the agent's behavioral interface. See Figure 13 for graphic representation of the implemented design pattern. Together, the methods which add triggers to the processing queues and those that process the triggers form the sensor-effector-chain as discussed in Chapter 3.

This design pattern shows a big advantage of an agent based approach to simulation. It is quite simple to expand and change the agent's behavior by adding or changing triggers and strategies. Therefore it is easy to test different implementation in order to find the best implementation.

GOALPROXY As stated above, the GoalProxy is the agent's behavioral interface. This class contains methods which can receive events from the environment and add Triggers to the goal queue and event queue of the agents. There is only one GoalProxy in the MAS. The GoalProxy contains a map of all the Schedulers of agents in the MAS.

SCHEDULERS For each agent there exists an object from the class Scheduler. This object is responsible for processing the triggers. Therefore, it contains two process trigger methods. Below the pseudo code of these methods is given. The processEventTrigger() processes the triggers on the event trigger queue by reactive behavior. This means that each trigger on this queue is only checked once. It does not matter whether it is possible or not to execute a strategy triggered by this trigger. The trigger is removed from the queue. The processGoalTrigger() processes the triggers on the goal event queue in a pro-reactive manner. Triggers on the goal queue are added at the back of the queue in case no strategy is executed for this trigger. A timer is activated which calls both those methods as long as the agent is active.

The design pattern of the pro-active method used to process triggers has been slightly adjusted for use in this program. The difference is the order in which the triggers in the trigger queue are tried to process. The design pattern checks for each strategy whether some trigger in the triggerQueue triggers this strategy. Therefore the order in which the triggers are executed (if applicable) is not the same as the order at which they are added to the trigger queue. Since the collection of strategies is arbitrary there is no way to control the order in which triggers are executed. It might be interesting to be able to control this. Therefore, the method has been adjusted in this program, resulting in that per trigger in the trigger Queue it is checked which strategy it triggers. By doing so, the triggers are processed if any strategy is triggered and applicable, in the same order as they are added to the trigger queue. Furthermore, if an applicable strategy is found, it is possible to control the order in which triggers should be processed, by changing the trigger queue into an ordered list and rank triggers.

Pseudo code of method which processes event triggers as proposed by Dastani and Testerink[8]:

```
public void processEventTrigger() {
   Trigger trigger = triggerqueue.remove();
   for(Strategy strategy : strategies) {
      if(!trigger.processed(context)&&
          strategy.triggeredBy(trigger))&&
          strategy.isApplicable(context)) {
             strategy.execute(context, trigger);
      }
   }
}
```

Pseudo code of method which processes goal triggers as used in this implementation:

```
public void processGoalTrigger() {
   for(int i = 0; i < triggerQueue.size(); i++) {
      Trigger trigger = triggerqueue.remove();
      for(Strategy strategy : strategies) {
         if(!trigger.processed(context)&&
            strategy.triggeredBy(trigger))&&
            strategy.isApplicable(context)) {
               strategy.execute(context, trigger);
         }
      }
      if(!trigger.processed(context)) {
         enqueue(trigger);
      }
   }
}
```

Pseudo code of method which processes goal triggers as proposed by Dastani and Testerink[8]:

```
public void processGoalTrigger() {
   for(Strategy strategy : strategies) {
      for(int i = 0; i < triggerQueue.size(); i++) {
         Trigger trigger = triggerqueue.remove();
         if(!trigger.processed(context)&&
            strategy.triggeredBy(trigger))&&
            strategy.isApplicable(context)) {
               strategy.execute(context, trigger);
         }
         if(!trigger.processed(context)) {
            enqueue(trigger);
```

```
                }
            }
        }
    }
```

CONTEXTINFORMATION The class ContextInformation is extended by the class DriverStatus. This class represents the belief base of a driver agent. It is assumed that the agent knows its own state. The context information consists of the name of the agent, the Rolling-Stock it is controlling, the limit of the current section, the limit of the next section, the name of the section it is subscribed to in case it is waiting for a section to be set free and a boolean that indicates whether the agent is waiting for a section to be set free. Furthermore, it can retrieve extra information from its RollingStock. This should be interpreted as if there is a display at which all parameters of the RollingStock are constantly visible to the agent. This was also discussed in Chapter 5. It thus needs to be viewed as an extension of the agent's belief base.

TRIGGERS In this MAS four different triggers occur. The first three triggers which are discussed below are event triggers, the latter is a goal trigger. For each event trigger an example of a situation in which these triggers will be pushed to an agent is shown in Figure 12. For each example a representation of the state of the railway system is given before and after the step is made by the rolling stock. The big gray dots are end nodes, the smaller gray dots are switches, the red dots are joints, the rolling stock is represented by blue dots and the edges represent both directions of the railway tracks. If the railway tracks of a section are gray the railway tracks are free, if the railway tracks of a section are orange the railway tracks are reserved and if the railway tracks are yellow the railway tracks of that section are occupied. Figure 12a shows what edges belong to what section.

The trigger *pass joint* is pushed by the environment whenever a RollingStock passes a joint in the disconnected mode. This is represented in Figure 12b. Before rolling stock *111111* has made its step it was located in section *6*. After the step is made rolling stock *111111* is located in section *4*. The trigger contains the limit of the section where the RollingStock just entered (i.e. section *4*), and it contains the limit of the section where the RollingStock will enter next (i.e. section *2*). Lastly, it contains the Location of the next joint (i.e. joint *2-4*) the RollingStock will pass.

The trigger *blocking* is pushed by the environment whenever the stopLocation of a RollingStock is within 10 meters from a Location at which it will enter an occupied section. The example in Figure 12c shows a step in which this trigger will be pushed. Rolling stock *222222* is located in section *2*, therefore section *2* is blocked for other rolling stock. Before the step is made rolling stock *111111* the end of the after running path of this rolling stock is not within 10 meters of the entrance point of section *2*, thus nothing happens. After the step is made section *2* is still occupied by rolling stock *222222*. However, the end of the after running path of rolling stock *111111* is now within 10 meters of the entrance point of section *2*. Therefore, the trigger *blocking* is pushed to rolling stock *111111*. This trigger does not contain any extra information. This trigger is an event trigger.

The trigger *release* is pushed by the environment to an agent whenever a RollingStock leaves a section to which that agent is subscribed. Figure 12d shows such a situation for rolling stock *111111* and rolling stock *222222*. Rolling stock *111111* cannot enter section *2*, because rolling stock *222222* is occupied that section before this step is made. Rolling stock *111111* is waiting for section *2* to be set free again and is subscribed to that section. After the step rolling stock *222222* is in section *1* and thus section *2* is set free again. The method which handles this checks whether a driver agent was subscribed to section *2*. This was the case for rolling stock *111111* thus a trigger *release* is pushed to rolling stock *111111*. This trigger does not contain any extra information about the environment.

The trigger *check goal speed* is only pushed when an agent is initialized in the disconnected mode. It contains the complete environment. This trigger is a goal trigger.

STRATEGIES There are four strategies in this MAS. See Figure 13.

The strategy *change limits* is triggered by triggerPassJoint. This strategy is always applicable, since it is assumed that drivers are never distracted from their job. If the strategy is executed, the currentLimit, nextLimit and nextJointLocation of the agent are set with the values of the equivalent parameters of the triggerPassJoint.

The strategy *blocking* is triggered by triggerBlocking. This strategy is always applicable, for the same reason as the changeLimitsStrategy. If executed the isBlocked boolean of the driver is set to true.

The strategy *release* is triggered by triggerRelease. It is applicable if the driver is blocked. The boolean isBlocked of the driver is set to false and the agent is removed from the subscription collection of the section it was subscribed to.

The strategy *check goal speed* is triggered by triggerCheckGoalSpeed. This strategy is always applicable also due to the assumption that a driver is never distracted. Figure 14 contains a flowchart of the method execute(). If executed, it is checked whether the goal speed should be changed. This depends on whether the driver is blocked, the limit of the current section and the limit of the next section and the location of the next joint. It is assumed that a driver is perfectly capable to estimate the exact braking distance and always wants to drive the maximum speed possible without violating any constraint at any time in the future. It might be interesting to expand this strategy with more research in order to create behavior which is more similar to the behavior of real drivers whom might sometimes want to violate constraints due to special circumstances. This is further discussed in Chapter 10.

7.3 MOVEMENT OF ROLLINGSTOCK IN ENVIRONMENT

If a rolling stock is initialized a timer is activated which represents the engine of the rolling stock. This timer calls the method doStep() repetitively. The size of the step equals the time interval between two calls of the method doStep(). By this parameter it is possible to determine the distance traveled in this time interval. The method calls the method moveInSimulation in Topolgy in which the actual movement and changes in the environment are handled. These methods are explained below. Figure 15 contains a flowchart of the method doStep() in the disconnected mode. Figure 16 contains the flowchart of the methods which handle the movement of the rolling stock in the evironment.

ROLLINGSTOCK As just stated, the method moveStep(), see Figure 15, calculates the distance traveled in the time difference as from the last time the method is called. It is taken into account whether the goal speed is equal to, higher than or lower than the current speed of the rolling stock. In case the rollingStock is accelerating or decelerating the speed is adjusted. Afterwards the method moveInSimulation() is called, afterwards which reserveSectionSimMode() is called.

The actual movement of the rolling stock needs to be handled by the environment due to the fact that neither the rolling stock nor the driver has control over the direction of the rolling stock. This is determined by the route.

TOPOLOGY The method moveInSimulation(), see Figure 16a, first sets the new location. If a joint is passed by the RollingStok in this step an event is pushed to GoalProxy to add an event change limits to the triggerEventQueue of the driver controlling the RollingStock. Afterwards the location where the brake distance ends is calculated and returned.

The method reserveSectionSimMode(), see Figure 16b, sets the section at which stop location is located at reserved if this section is not the same section as where the RollingStock is located. Subsequently the first joint on the route after the stop Location is calculated. It is checked whether the distance between the stop Location and the joint right after is less than 10 meters. And, if the next section is occupied an event is pushed to GoalProxy to add an event blocking to the triggerEventQueue of the driver controlling the RollingStock.

ADDING ROLLING STOCK The previous chapter explained that only the yard Amersfoort was taken into account in this thesis. By doing so it seems necessary to add new rolling stock during the disconnected mode in order to maintain a accurate simulation. If the whole railway infrastructure was represented, rolling stock with origin Utrecht would arrive in the yard Amersfoort after for example one minute in the disconnected mode. Methods were implemented which inject rolling at their planned locations during the disconnected mode. These methods are not further explained since this is just a temporary solution.

7.4 STARTING POINT

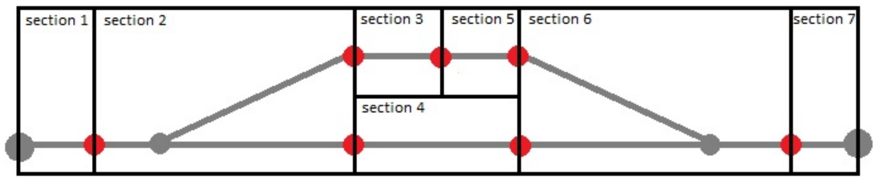
The problem for which a solution is proposed in this thesis is to find a way to predict the future locations of the trains in the railway system. Therefore, this MAS is developed in order to be able to simulate the behavior of the system in a very simple manner. However, the agents should not be initialized as *tabula rasa*. The agent should all immediately behave as if they were agents long ago. The simulation should be started with the current state of the railway system. This basically means that, ideally, all rolling stock should have exactly the same speed and location as in the real system and the drivers should act as if they where agent before and therefore have all parameters which are set by triggers up to date. In that case they would react accurate right from the first moment in the simulation. Therefore, data from this real railway system should be endowed to the agents at the start of the simulation. The following chapter explains the methods which handle this transition

7.5 CONCLUSION

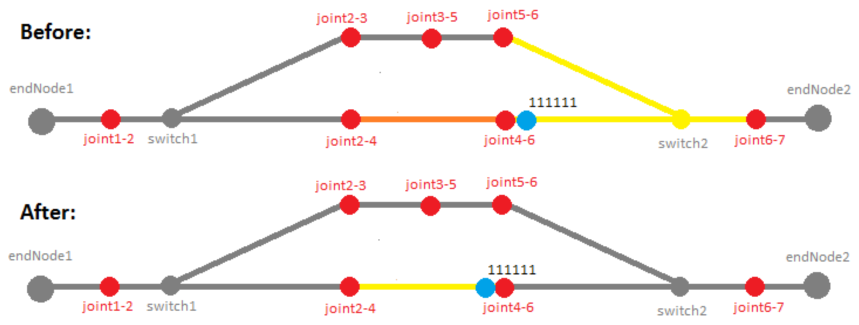
This chapter started with a theoretical definition of the driver agents. The reactive ability of a driver agent is to observe joint passes, blockings and releases of sections. Their pro-active ability is to check for an accurate goal speed. They interact with the environment via triggers and executable strategies. Their dependence relation is depending on the viewpoint from their individual goals or the goal of the system either independent or mutual dependent. The dependency is mutually believed by all agents since they should cooperate.

This chapter answered the sub question: how is the disconnected mode of the tool which can predict the future locations of the trains implemented? The drivers are the only agents in this implementation. A design pattern for agents in object-oriented programming is used to implement the agents. Triggers represent the perception of the agents and execution of strategies represent the action of agents. The triggers of driver agents are: pass joint, blocking, release and check goal speed. The strategies are: change limits, blocking, release and check goal speed. The rolling stock handles the movement of the rolling stock in the environment. The step size depends on the time step, the current speed and the goal speed. Triggers are pushed if necessary. This design pattern showed a big advantage of agent-based simulation. The behavior of the agents and thereby the behavior of the whole system can easily be adapted by changing or adding triggers and strategies. This advantage perfectly fits in with the reasons given in Chapter 2 for choosing an agent-based approach. With a complex central solution it would be much harder, for example, to define which parameters should be adapted in order to change the representation of the behavior of the drivers of the original systems. Hence, improving a model and therefore necessarily changing the executable model becomes less complex.

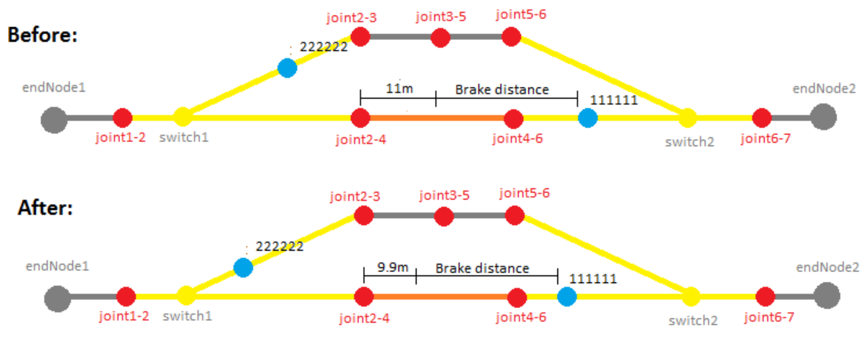
Finally, this chapter explained that it is necessary to endow agents with real data at the moment of their initialization in the disconnected mode. The behavior of the agent should be accurate from the beginning.



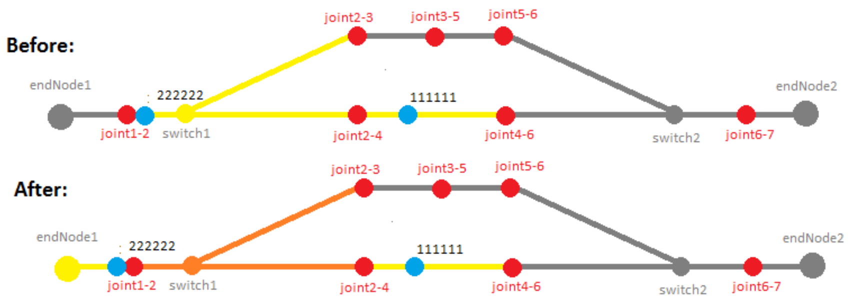
(a) Devision of sections



(b) Trigger pass joint



(c) Trigger blocking



(d) Trigger release

Figure 12: Examples of steps made by rolling stock in which triggers occur. In case printed in black and white colors might not be distinguishable

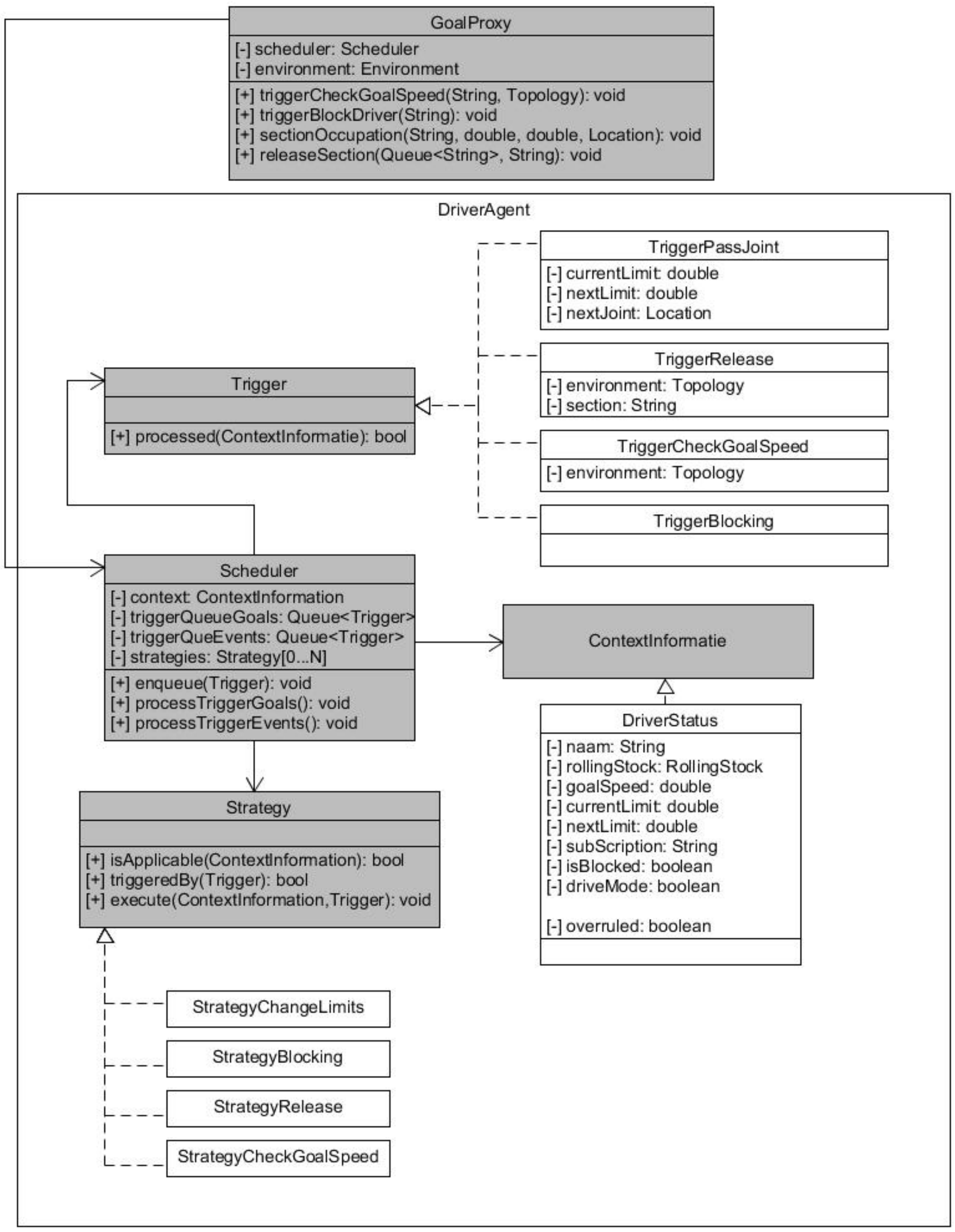

GoalProxy
[-] scheduler: Scheduler
[-] environment: Environment
[+] triggerCheckGoalSpeed(String, Topology): void
[+] triggerBlockDriver(String): void
[+] sectionOccupation(String, double, double, Location): void
[+] releaseSection(Queue<String>, String): void
DriverAgent
TriggerPassJoint
[-] currentLimit: double
[-] nextLimit: double
[-] nextJoint: Location
TriggerRelease
[-] environment: Topology
[-] section: String
TriggerCheckGoalSpeed
[-] environment: Topology
TriggerBlocking
Trigger
[+] processed(ContextInformatie): bool
Scheduler
[-] context: ContextInformation
[-] triggerQueueGoals: Queue<Trigger>
[-] triggerQueEvents: Queue<Trigger>
[-] strategies: Strategy[0...N]
[+] enqueue(Trigger): void
[+] processTriggerGoals(): void
[+] processTriggerEvents(): void
ContextInformatie
DriverStatus
[-] naam: String
[-] rollingStock: RollingStock
[-] goalSpeed: double
[-] currentLimit: double
[-] nextLimit: double
[-] subScription: String
[-] isBlocked: boolean
[-] driveMode: boolean
[-] overruled: boolean
Strategy
[+] isApplicable(ContextInformation): bool
[+] triggeredBy(Trigger): bool
[+] execute(ContextInformation,Trigger): void
StrategyChangeLimits
StrategyBlocking
StrategyRelease
StrategyCheckGoalSpeed

Figure 13: UML of driver agent

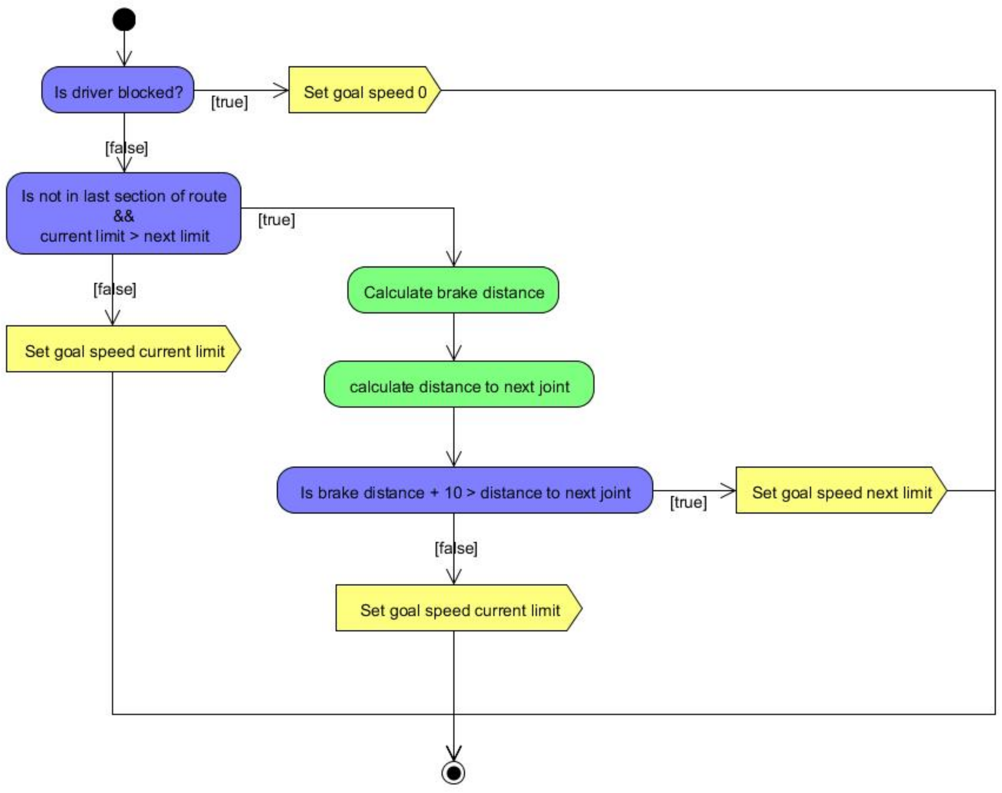

Is driver blocked?
[true]
Set goal speed 0
[false]
Is not in last section of route && current limit > next limit
[true]
[false]
Set goal speed current limit
Calculate brake distance
calculate distance to next joint
Is brake distance + 10 > distance to next joint
[true]
Set goal speed next limit
[false]
Set goal speed current limit

Figure 14: Flowchart of method execute() of checkGoalSpeedStrategy

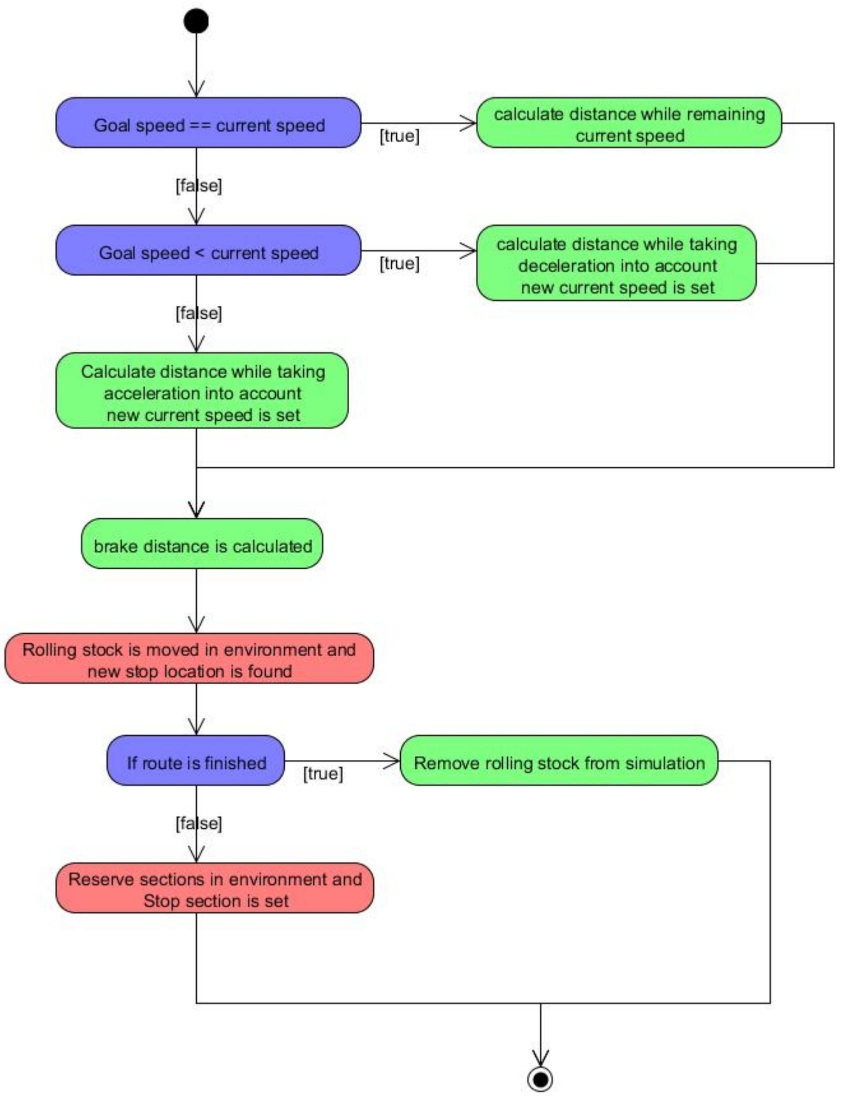
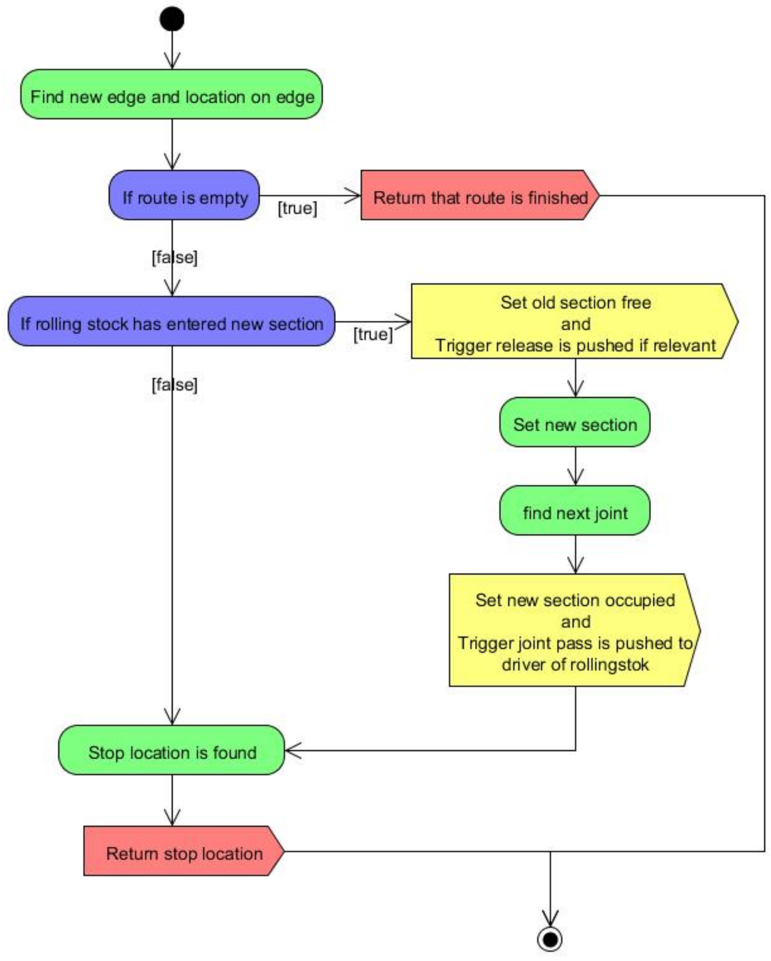
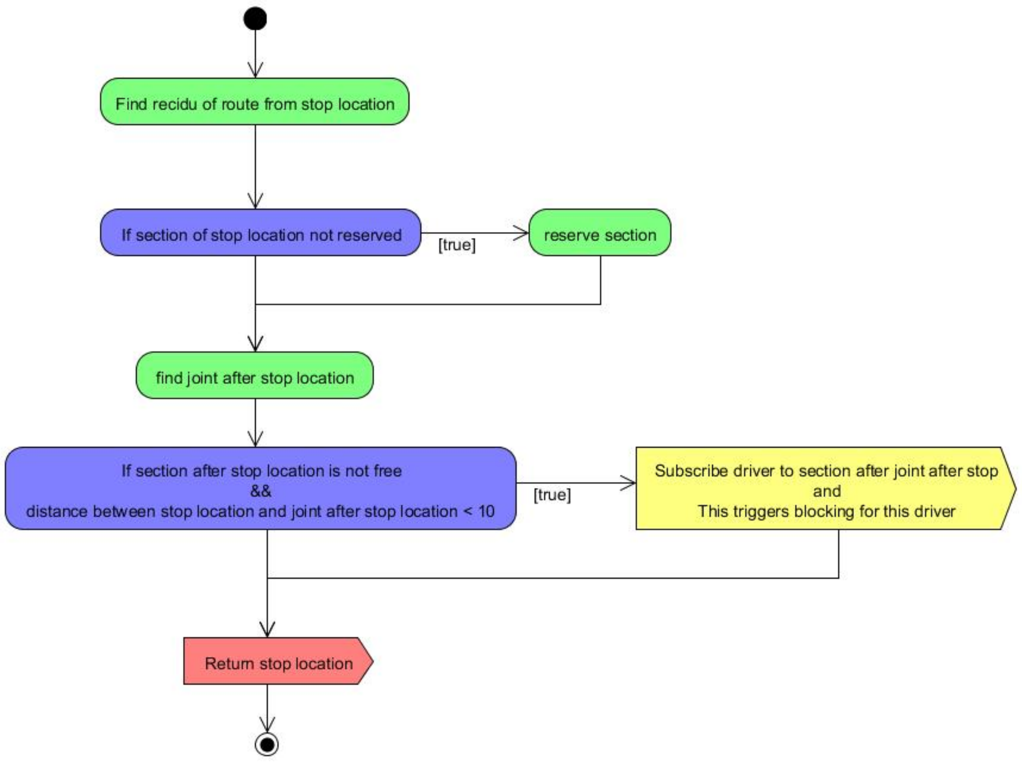

Goal speed == current speed
[true]
calculate distance while remaining current speed
[false]
Goal speed < current speed
[true]
calculate distance while taking deceleration into account new current speed is set
[false]
Calculate distance while taking acceleration into account new current speed is set
brake distance is calculated
Rolling stock is moved in environment and new stop location is found
If route is finished
[true]
Remove rolling stock from simulation
[false]
Reserve sections in environment and Stop section is set

Figure 15: Flowchart of method doStep() in disconnected mode


Find new edge and location on edge
If route is empty
[true]
Return that route is finished
[false]
If rolling stock has entered new section
[true]
Set old section free and Trigger release is pushed if relevant
Set new section
find next joint
Set new section occupied and Trigger joint pass is pushed to driver of rollingstok
[false]
Stop location is found
Return stop location

(a) Method which handles movement of rolling stock in disconnected mode


Find recidu of route from stop location
If section of stop location not reserved
[true]
reserve section
find joint after stop location
If section after stop location is not free && distance between stop location and joint after stop location < 10
[true]
Subscribe driver to section after joint after stop and This triggers blocking for this driver
Return stop location

(b) Method which handles sectioin reservation in disconnected mode

Figure 16: Methods which handles movement and reservation in disconnected mode

8 TRANSITION

The previous chapter defined what an agent should be capable of at the start of a simulation run. This chapter answers the sub question: how is the transition between the connected mode and the disconnected mode implemented in the tool which can predict the future locations of the trains implemented? Furthermore, it is explained what data should be kept up to date in the connected mode in order to be able to start the disconnected mode at any given time.

In the first section it is explained why the environment of the MAS is used in the connected mode as well. The second section will explain what data is necessary to uphold in order to be able to start a simulation run at any moment in time from the connected mode. The third section will explain the methods which are called when a simulation is started. The chapter ends with a concluding section.

8.1 USE OF ENVIRONMENT IN BOTH MODES

The connected mode and the disconnected mode both make use of the same environment. This is the environment as described in Chapter 6. In the connected mode the rolling stock is controlled by the live-feed of data from the real railway system. In the disconnected mode the rolling stock is controlled by the agents from the MAS developed for this program. By using this technique it is quite easy to switch from one mode to another. In the connected mode it is as if the rolling stock is controlled by the drivers of the real rolling stock outside. Therefore it does not make sense to have agents which represent the drivers in the connected mode.

8.2 DATA TO UPHOLD IN CONNECTED MODE

As explained at the end of the previous chapter, the agents should not be initialized with a blank belief base. To be able to make a good prediction the agents should all be in a state as if they where agents before and have processed relevant triggers in the past. This should result in a simulation that starts in the exact same state as the actual system. It is explained in Chapter 2 that this is very important in order to make a good prediction of the future locations of the rolling stock. In Chapter 10 it is discussed whether the data used is exact enough.

For all parameters which are set directly or indirectly by strategies of the agents it should be checked what their accurate value would be. The directly set parameters are: current limit, next limit, next joint location, goal speed, whether or not the driver is waiting for a section to be released and, in case it is waiting, the name of the section the agent is subscribed to.

Furthermore, in an ideal situation the exact velocity of a rolling stock should be directly set by the data of the real situation. Or it should be derived from the time difference and distance between two joint passes. However, due to the fact that there is no data available to retain the exact velocity of a rolling stock and because the real length of the sections is not known for each section, the speed of the rolling stock has to be calculated from the directly set parameters as well. Whether or not this is a problem is discussed in Chapter 10.

To be able to correctly set all these parameters the following data has to be uphold in the connected mode: The current location of a rolling stock, the plan rules and route of the rolling stock. Finally, if two rolling stocks will cross each others paths in the future it should be known which rolling stock has priority over the other. Which rolling stock has priority partly depends on which rolling stock reserved a section first, and therefore it is necessary to uphold which sections are reserved for which rolling stock in the connected mode. Because doing so can determine which rolling stock has priority.

8.3 METHODS WHICH HANDLE TRANSITION

There are two methods responsible for the transition from the connected mode to the disconnected mode. The first method simply takes a snap shot of the state of the system in the connected mode. The second method sets the parameters, current limit, next limit and next joint location. Furthermore, by initializing the agent a method is called which checks whether or not a driver has to brake for some reason and therefore sets the other parameters: speed, goal speed, whether it is blocked or not and if so, by what section. The following paragraphs explain and discuss these methods.

TAKE SNAP SHOT Due to the fact that the connected mode will maintain running, it is not enough to just set a boolean isSimulating to true and adding a driver agent to the rolling stock. A deep copy has to be made of all the rolling stock currently active in the connected mode. By doing so it remains possible to observe the rolling stock in both modes. A copy of all the sections has to be made as well. However, the sections parameters reservedFor and occupiedBy will not be copied from the connected mode due to the fact that the speed cannot be derived accurately from the data of VIEW. These parameters will be set later in the transition phase. This will be further explained later.

The parameters set in the copies of the rolling stock are: name, length, max speed, plan rule, route and location. By setting the route it is checked whether the rolling stock is currently situated on its route. If not, the rolling stock is not added to the simulation. See Figure 17 for flowchart of method.

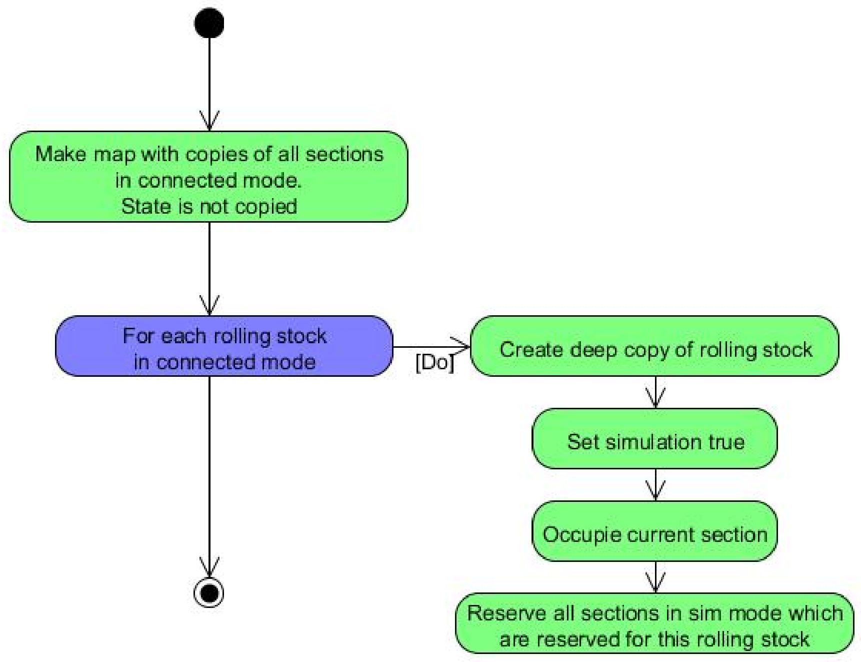

Make map with copies of all sections in connected mode. State is not copied
For each rolling stock in connected mode
[Do]
Create deep copy of rolling stock
Set simulation true
Occupie current section
Reserve all sections in sim mode which are reserved for this rolling stock

Figure 17: Method to take snap shot of connected mode

INITIALIZE AGENTS For each rolling stock which is copied to the disconnected mode the parameters current limit, next limit and next joint are set. If the rolling stock is already at the end of its route no next joint can be found. Therefore, the rolling stock is deleted from the disconnected mode and the driver agent is not initialized. A new DriverAgent is initialized, if there is a next joint. The DriverAgent is set as the driver of this rolling stock and a new Scheduler is initialized to create an agent. This is described by the design pattern as discussed in Chapter 7. See Figure 18 for flowchart of method.

CHECK FOR BLOCK AND BRAKE The correct stop location is calculated at first, thereby taken into account that the current limit might not be the accurate speed due to blocking. Thereafter it is checked whether the rolling stock is blocked.

If blocked, timeToBreak is called with goalSpeed 0 in RollingStock. This method calculates the maximum possible speed in order to not ignore any constraints due to the blocking ahead. As already explained above, due to limited access to exact data about joint loca-

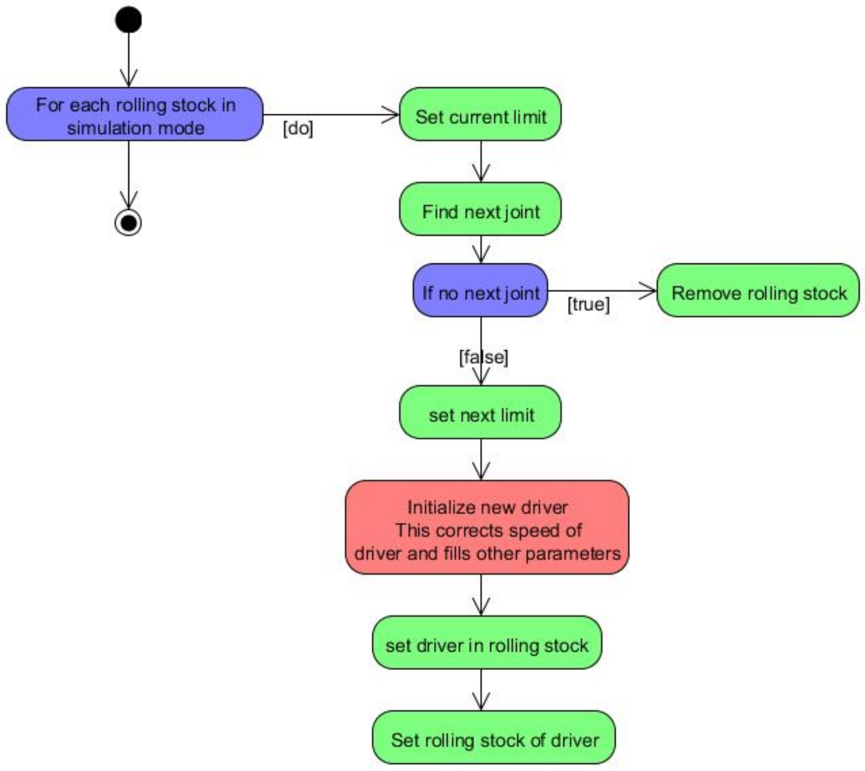

For each rolling stock in simulation mode
[do]
Set current limit
Find next joint
If no next joint
[true]
Remove rolling stock
[false]
set next limit
Initialize new driver
This corrects speed of driver and fills other parameters
set driver in rolling stock
Set rolling stock of driver

Figure 18: Method to initialize agents in disconnected mode

tions and actual speed of rolling stock it is impossible to estimate the right speed in the connected mode. Therefore, it has to be calculated in the transition mode.

This method manipulates the speed. After changing the speed it is checked to see if this new speed is not too high to enter the next section. If this is not the case the agent is subscribed to the blocked section, the goal speed is set to 0 and true is returned. Otherwise the speed is manipulated to be able to achieve the next limit on time, the goal speed is set to next limit and false is returned.

If the section after stop location is not blocked, timeToBreak is called to check whether the speed should be manipulated in order to be able to achieve the next limit on time. The goal speed is set to the next limit and false is returned. If the rolling stock can maintain the current limit as its current speed the goal speed is set to current limit and false is returned. See Figure 19 for flowchart of method.

8.4 CONCLUSION

This chapter explained what should happen in order to start a simulation. It also answered the following sub question: how is the transition between the connected mode and the disconnected mode imple-

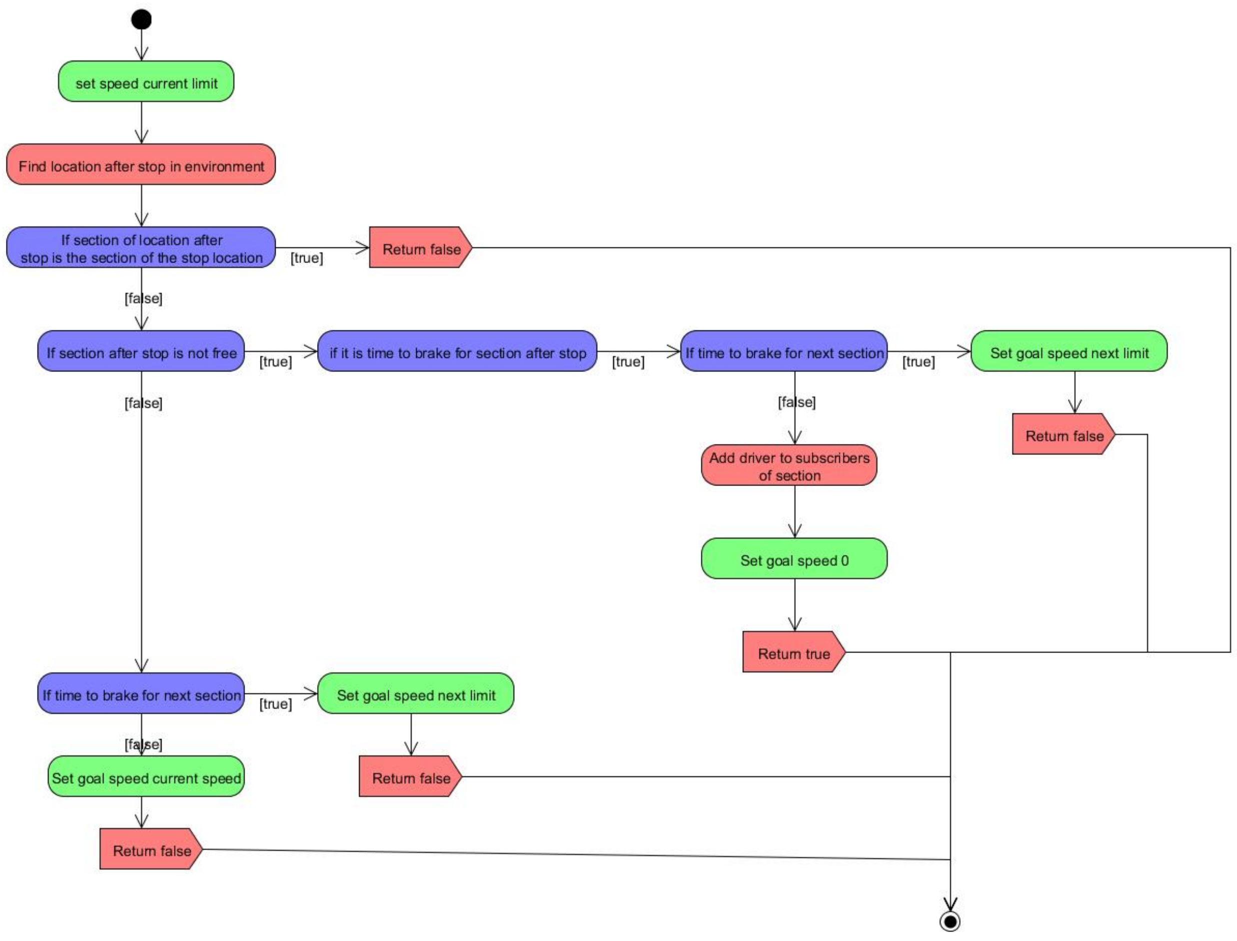

set speed current limit
Find location after stop in environment
If section of location after stop is the section of the stop location
[true]
Return false
[false]
If section after stop is not free
[true]
if it is time to brake for section after stop
[true]
If time to brake for next section
[true]
Set goal speed next limit
Return false
[false]
Add driver to subscribers of section
Set goal speed 0
Return true
[false]
If time to brake for next section
[true]
Set goal speed next limit
Return false
[false]
Set goal speed current speed
Return false

Figure 19: Method to check for blocking and set speed and goal speed

mented in the tool which can predict the future locations of the trains implemented? Two methods are necessary to initialize the agents in the right order. Firstly, a snap shot is taken of the current state of the model in the connected mode. Secondly, the agents are initialized with the accurate data and the parameters, which are normally set by strategies, are set. The necessary data consists of the actual location of the rolling stock, the route left to travel, the plan rules and the reserved sections.

9 CONNECTED MODE

The previous two chapters show that the following data is necessary to uphold in the connected mode to be able to start the disconnected mode at any time: the actual location of the rolling stock, the route left to travel, the plan rules and the reserved sections. The following sub question is answered: how is the connected mode of the tool which can predict the future locations of the trains implemented? Furthermore the data which is collected from the real system is evaluated.

The first section explains what data is available and it is evaluated. The second section explains the general idea of the connected mode. The third section discusses the methods to process the available data. The fourth section discusses the methods which are responsible for the actual movement of the rolling stock in the connected mode. The chapter finishes with a concluding section.

9.1 AVAILABLE DATA

The data used in the connected mode is data from VIEW. The connected mode is subscribed to events which contain information about at what time what journey number (and the rolling stock executing this journey) passes what joint. Furthermore the connected mode should be subscribed to events in which it is stated which plan rules are set for a specific journey number.

It was assumed beforehand that the data is consequent and does not contain any errors. Unfortunately, during the research it became clear this was not the case. Especially in the events in which a rolling stock changes sections, errors occur. Such an event should indicate what joint is passed by that rolling stock on that moment, because joints are the connection between two sections. However, VIEW is not consequent in which sections are connected with other sections. The events sometimes state that a change of sections is made which does not exist. If you, for example, take Figure 20 as the railway infrastructure, an event might occur which states that a rolling stock went from section 2 to section 6 even though there exists no joint2-6. A pattern in these errors is not found. It is out of the scope of this thesis to find the origin of these errors. The program is implemented in a way it can handle these errors.

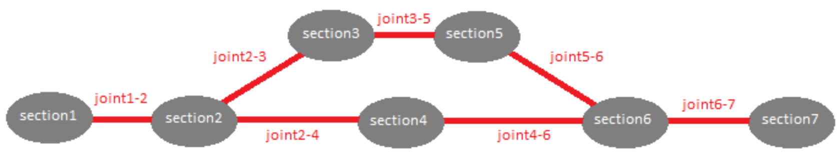


Figure 20: Section graph

9.2 PROCESSING EVENTS

An event handler handles the events which originate from VIEW, initializes new RollingStock if necessary and removes RollingStock which are no longer active.

For this particular implementation some adaptations are made in order to make testing possible. The program is not connected to the on-line data events from VIEW. A set of joint pass events of two hours is recorded and saved in a CSV-file. The set routes from the same time frame are stored in a map. The method which processes the events can be used both for this implementation as well as the implementation in which the program would be subscribed to the on-line data events from VIEW. The method to process the set route events should be developed. This is further discussed in Chapter 10.

9.3 METHODS TO PROCESS EVENTS

The following paragraphs explain and discuss the methods which are used to process the events.

METHOD TO PROCESS JOINT PASS EVENTS Every pushed joint pass event is processed. Below it is explained what happens if this method is called. Figure 21 contains a flowchart of this method. The parameter event contains journey number, joint pass and plan rule.

Any time an event arrives for a certain rolling stock that is already in the connected mode the location of the rolling stock is relocated. The new location is set to the beginning of the edge which origin is that the joint the rolling stock just passed. The speed is calculated by the actual distance traveled since the last event, divided by the time difference between those two events. Every time when an event for this rolling stock is pushed, the plan rule is set. This is to make sure that the last plan rule is known in case of a disconnection.

A new rolling stock is initialized when a joint pass event of a rolling stock arrives, which is not already in the connected mode. The location is calculated in the same way as with a known rolling stock. The speed of the rolling stock is assumed to be the minimum of the maximum allowed speed in the current section and the maximum speed of the rolling stock. The plan rule is set and the rolling stock is activated. This activation will handle the movement of the rolling stock and is therefore further explained in the next section.

In case a joint pass event arrives whereby the pass is not in the focus area of this program or does not exist at all it is checked whether it is about a rolling stock already in the connected mode. If so, this rolling stock is removed. If the rolling stock is not in the connected mode nothing happens.

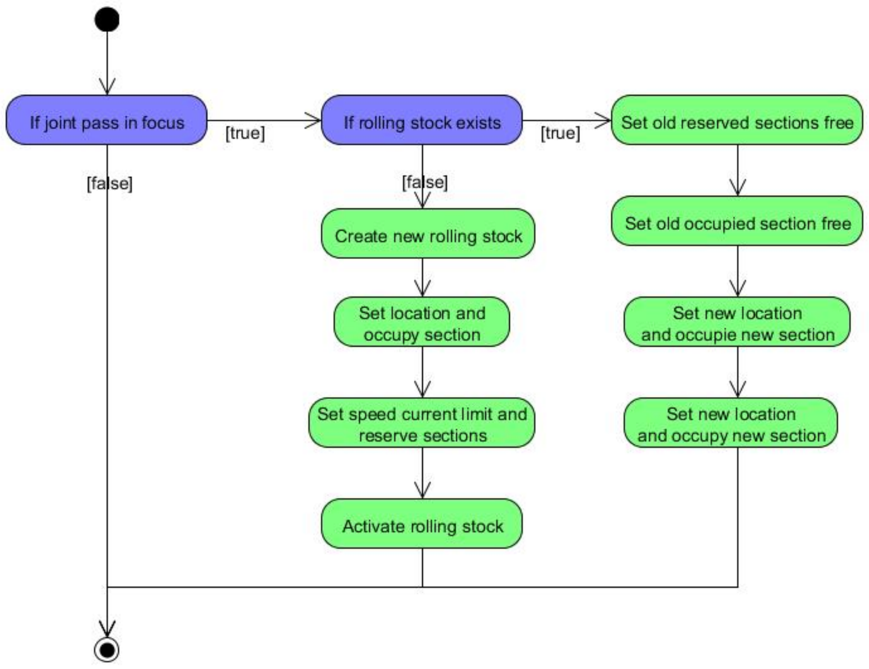

If joint pass in focus
[true]
If rolling stock exists
[true]
Set old reserved sections free
[false]
[false]
Create new rolling stock
Set old occupied section free
Set location and occupy section
Set new location and occupie new section
Set speed current limit and reserve sections
Set new location and occupy new section
Activate rolling stock

Figure 21: Method to process joint pass event from VIEW

RESERVING SECTIONS The protocol for reserving sections deserves some explanation. Whenever a new RollingStock is initialized the sections which are part of the after-running path are reserved one by one until no sections are left or a section is not free. The braking distance is calculated by the assumed speed. It is always assumed the rolling stock will retain the same speed.

Every time a new event arrives for a rolling stock which is already active in the connected mode, the old reserved sections are set free. And the new reserved sections are determined in the same manner as with a new rolling stock. This has to be done because the rolling stock's speed can be much lower than expected, making the braking distance much shorter and eliminating the need for sections at the end having to be reserved. If a rolling stock is removed from the connected mode the reserved sections of that rolling stock are set free too.

Furthermore, a section can also be set free if a rolling stock is relocated at a section which was previously assumed to be reserved for another rolling stock. In that case the section is set occupied by the rolling stock which is relocated in that section. At the same time, the sections which are reserved for the other rolling stock, and which are behind the occupied section on the route of the rolling stock, are set free again. Of course, it may be possible to see whether other rolling stocks, having previously been blocked by this rolling stock, are now capable of reserving more sections. However this is not implemented in this program.

9.4 MOVEMENT OF ROLLING STOCK

In the connected mode there is one certainty about the location of the rolling stock. Consider a rolling stock passed a joint at time t. At time t+x the rolling stock is at least on the edge after the passed joint. A speed can be assumed as explained in the previous section. Therefore, it can be assumed where a rolling stock is at time t + x. However, a rolling stock can never be assumed to be on a next edge since it would pass a node which either is a switch or a joint. In case it is a switch it is a decision point, due to the implementation of the infrastructure it is not possible to know which direction the rolling stock will take. In case it is a joint this means a new joint pass event should occur before the rolling stock will pass this joint.

The following paragraphs will discuss the most important methods to this movement in detail.

DO STEP METHOD By activating a rolling stock a timer is activated which calls the method doStep every 1/10 second. This method is very simple in case the rolling stock is in the connected mode. The traveled distance in 1/10 second is calculated and the breaking distance is calculated. Secondly, a the method moveInConnectedStage in Topology is called which returns the reserved sections for this rolling stock. The actual movement of the rolling stock is handled by the method moveInConnectedMode() in Topology. See Figure 22 for a flowchart of the method.

MOVE IN CONNECTED STAGE METHOD Checked is whether the remainder of the edge is long enough to do the complete step. If so, the new location on the edge is the step added to the old location. Otherwise, the new location would be set at the end of the edge. Thereafter, the remainder of the route is found by means of the edge the rolling stock is on and the plan rule. This is necessary to be able to reserve the correct sections for this rolling stock. The queue of reserved sections is returned. See Figure 23 for a flowchart of the method.

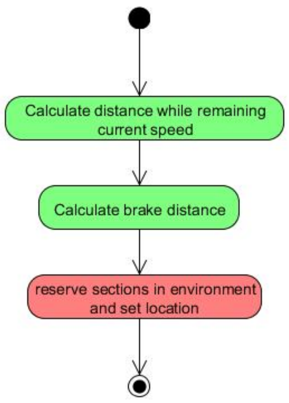

Calculate distance while remaining current speed
Calculate brake distance
reserve sections in environment and set location

Figure 22: Flowchart of method to do step in connected mode

9.5 CONCLUSION

This chapter explains how the necessary data for starting a simulation is collected. This is also the answer to the following sub question: how is the connected mode of the tool which can predict the future locations of the trains implemented? One must realize that in this mode agents do not exist. It is the rolling stock which is connected to the live feed in the connected mode. It should be seen as if the rolling stock is controlled by the actual drivers. In the disconnected mode the rolling stock is controlled by the driver agents.

The rolling stock is connected to the live-feed via the following two methods: a method to process pass joint events and a method to process change plan rules events. These methods correct the location and plan rules of the rolling stock. And delete the rolling stock if necessary.

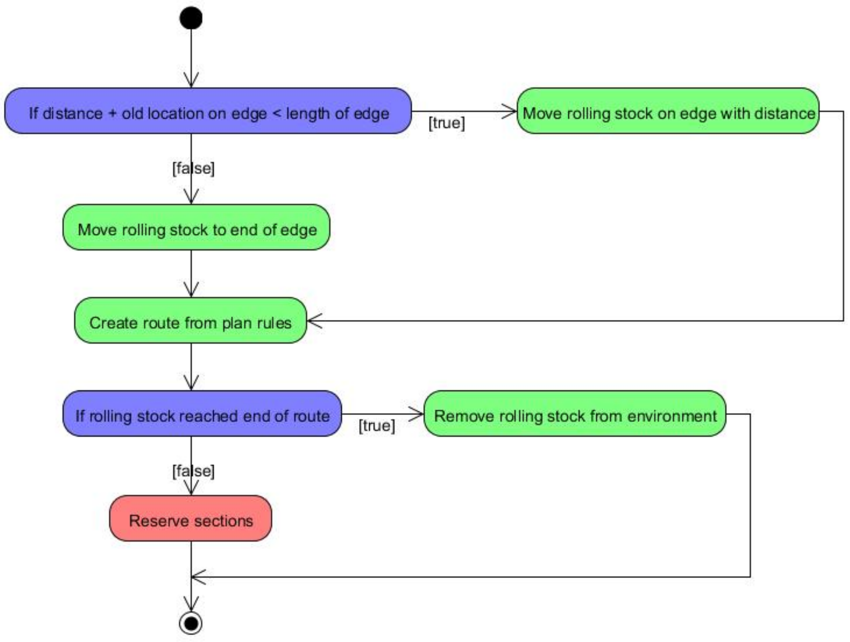

If distance + old location on edge < length of edge
[true]
Move rolling stock on edge with distance
[false]
Move rolling stock to end of edge
Create route from plan rules
If rolling stock reached end of route
[true]
Remove rolling stock from environment
[false]
Reserve sections

Figure 23: Flowchart of method to move rolling stock in environment in connected mode

Part III

DISCUSSION

10 EVALUATION AND RECOMMENDATIONS

This chapter will evaluate the prototype of the program which is developed for this research project. By doing so an answer to the research question of this thesis can be formulated. This research project contributes to the development of a tool that supports controllers by predicting the future location of trains in the railway system by simulating railway scenarios. In order to predict future locations and to simulate railway scenarios it is necessary to understand the underlying technologies, the data of the railway system used to rightly start the simulation and the necessary aspects of the railway for railway scenarios. Therefore, the research question of this thesis is: what aspects of the state of the trains in the Dutch railway network and aspects of the network itself are relevant, and what technologies are necessary to make an agent-based simulation tool which is able to correctly simulate railway scenarios which predict (based on the current location of the trains, other relevant aspects of the state of the trains and the railway network) the future locations of the trains in the Dutch railway network?

This chapter consists of four sections. In order to find an answer to the research question and to be able to develop a prototype of the tool, seven sub questions have been developed which are answered in the previous chapters. Furthermore, a literature search has been conducted in order to obtain the latest findings and theories related to this problem and to support the choice for an agent-based approach in order to solve this problem. The first section of this chapter will give a summary of the interesting findings in the problem related literature. Furthermore it will recapitulate the answers to the sub questions given throughout this thesis.

As explained in Chapter 3, the program used is developed according to the model process of Siegfried[16]. The thesis' sub questions correspond with different steps of the process model. This thesis fully conducted the first three steps. The result of the first two steps, which is the problem description to develop a tool which can predict future locations of trains, is described in Chapter 1. The conceptual model, which is step three, is described in Chapter 4. It was stated that step four to six should be iterated. The first iteration has been conducted and described in the previous chapters. In step four the model is simplified without loosing necessary aspects. Step five translates this simplified model into an executable model. In step six the model is executed and the results are interpreted. The evaluation of this first iteration will take place in section two of this chapter.

In the third section of this chapter the research question is answered. Section four concludes whether it is possible to develop a tool which can predict the future locations of trains in the railway system. It also also looks into whether such a tool can be of interest to ProRail and it formulates recommendations for further research and development of the tool.

10.1 FINDINGS IN PROBLEM RELATED LITERATURE AND ANSWERS TO SUB QUESTIONS

As stated, this section gives a summary of interesting findings in the problem-related literature. This is described in the first paragraph. Thereafter, the sub questions are discussed and the answers are recapitulated in different paragraphs.

STRUCTURED PROBLEM DESCRIPTION The problem to which this thesis contributes, is the fact that it is very hard to predict the future locations of trains in the railway system. A railway system without any delays is wanted. However, due to its intensive use and the limitations to physical expansion, other solutions have to be found in order the reduce the amount of delays. The literature gave proof for the use of the actual state of the railway system as a starting point of the predictions. The current situation does have an influence on the outcome of the prediction. However, until now only tools exist which make predictions based on historical data. A huge drawback is that by making use of this data only results of scenarios which occur more often can be predicted. It was agreed on that an agent-based approach might be able to solve this. The benefit of an agent-based approach is that results of new scenarios can also be predicted.

The other benefits of using an agent-based approach are: it enables solving the problem in a natural way since the active entities can have a local perspective, agents are perfect to model a heterogeneous system since all agents can be developed with their own architecture, state and behavior. It is possible to talk about the agent-based system on a natural level, which makes it comprehensible. Lastly, it was stated that this approach is of special interest to traffic management.

In Chapter 7 a new benefit of an agent-based approach was introduced. The design pattern as proposed by Dastani and Testerink shows that it is rather simple to change or expand the behavior of an agent by adding or changing triggers and strategies. These benefits show why it is interesting for ProRail to opt for a tool that makes use of an agent-based approach. Due to the fact that this tool can handle predictions of new scenarios, it is still relevant when rare incidents occur or when the railway's infrastructure is changed. The fact that the behavior of an agent can easily be changed or expanded is also interesting for ProRail, due to the fact that it is simple to change the strategies or triggers if changes are made in the protocols that drivers have to follow. In brief, a tool that is agent-based is adjustable in a natural way and can be changed according to changes in the real system without the necessity to develop a complete new simulation.

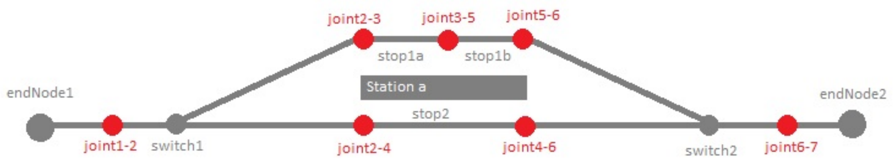
INFRASTRUCTURE AND RULES The sub question is: what elements does the infrastructure of the railway system consists of, and what rules apply for trains and drivers in the railway system?

Figure 24b is an example of an infrastructure as can be found outside. The infrastructure has three viewpoints from which this actual infrastructure can be represented. Firstly, there is the schematic viewpoint in which the topology is represented as a graph in which the edges are railway tracks and the nodes are switches and buffer stops see Figure 24b. The second viewpoint is the safety viewpoint and is a graph in which the edges are the joints and the nodes are the sections Figure 24c. The last viewpoint is the macro topology seeFigure 24d. This is a graph in which the edges are the free tracks and the nodes are the control points. The most important rules which have to be obeyed is that rolling stock can never go faster than the maximum speed of the current section and only one rolling stock at a time is allowed in a section.

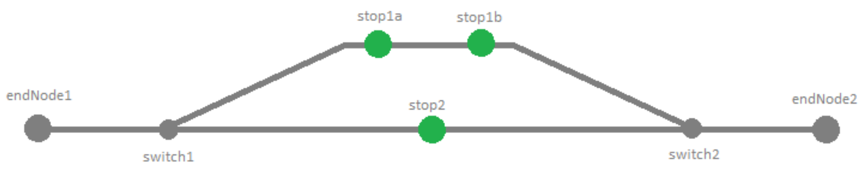
TASKS AND RESPONSIBILITIES The sub question is: which actors of the railway system (i.e. the original system) have to be included in the agent-based simulation to make sure the result of the simulation of the railway scenarios does not significantly deviate from the scenarios in the original system, and what are the responsibilities of these actors?

The driver of the rolling stock controls the speed of the rolling stock and has the following goal: arrive on time at the scheduled stops without violating any constraints. The local traffic controller is responsible for the safety and has to set the routes of the rolling stocks. The regional traffic controller has the task to keep delays to a minimum. The national traffic controllers are responsible for the general traffic flow.

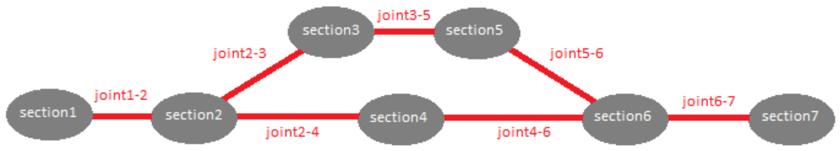
CONDUCTED STEPS OF MODEL PROCESS The previous two sub questions together form the conceptual model developed of the railway system. This is step three of the process model of Siegfried. As stated before only the first iteration of the other steps of the program is conducted. Therefore, the answers to the following sub questions should be seen as a first iterations as well. These answers, together with the evaluation of the program run will result in an answer to the main question. Depending what the answer to the main question is, it might result in the answers to the latter sub questions having to be



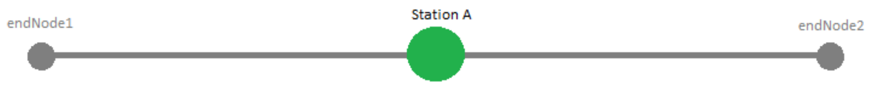
(a) Railway infrastructure



(b) Schematic viewpoint



(c) Safety viewpoint



(d) Macro topology

Figure 24: Viewpoints of railway infrastructure

reformulated. These actions are formulated as recommendations for further research.

SIMPLEST MODEL POSSIBLE The sub question is: what is the simplest agent-based model possible of which a simulation of the railway scenarios would not significantly deviate from the scenarios which occur in the original system?

The railway infrastructure is represented as a graph in which the schematic viewpoint and the safety viewpoint are integrated. Signals are not represented in the model, however their goal to allow at most one section in a block to keep rolling stock at a safe distance from each other is preserved. The rolling stock and journey numbers are united. The rolling stock is represented in quite the same way as in the real situation. Furthermore, it contains a display with its current speed, exact location on the graph, to travel route and plan rules. The length of a rolling stock is not taken into account and its maximum speed is set to 200 km/h. A driver has only one goal. This is to never violate a speed constraint. The knowledge of the driver contains its own state, the limit of the current section, the limit of the next section and it can check the display of the rolling stock any time. Controllers are not represented in this model. Therefore the routes are fixed.

Furthermore it is important to notice that due to the focus on the yard of Amersfoort it was decided to inject new rolling stock into the simulation after the running time was started to represent the rolling stock which would arrive at the yard during the simulation run.

ENVIRONMENT OF MODEL The sub question is: how to represent the infrastructure and the rolling stock of the railway system in the environment of the multi-agent system?

The railway infrastructure is represented as a directed graph, see Figure 25. The edges represent the railway. For each railway track there are two edges one for each direction. The nodes represent the switches, endNodes and joints. A section is not directly present in the graph. However, for each edge it is known to which section it belongs. The rolling stock is represented as objects in the environment since it are identifiable discrete entities in the environment without autonomous behavior.

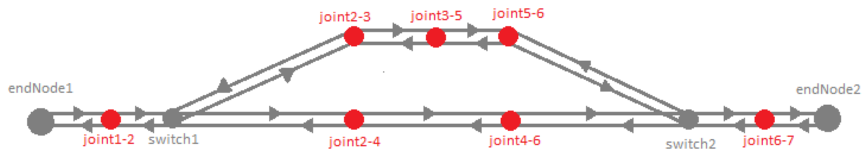


Figure 25: Graph as implemented

DISCONNECTED MODE The sub question is: how is the disconnected mode of the tool which can predict the future locations of the trains implemented?

The drivers are the only agents in this implementation. A design pattern of Dastani and Testerink[8] for agents in object-oriented programming is used to implement the agents. Triggers represent the perception of the agents and execution of strategies represent the action of agents. The triggers of driver agents are: pass joint, blocking, release and check goal speed. The strategies are: change limits, blocking, release and check goal speed. The rolling stock handles the movement of the rolling stock in the environment. The distance traveled in one step depends on the time step, the current speed and the goal speed. Triggers are pushed if necessary.

TRANSITION PHASE The sub question is: how is the transition between the connected mode and the disconnected mode implemented in the tool which can predict the future locations of the trains implemented?

The agents should not be initialized as *tabula rasa* at the start of the simulation, since the behavior of the agent should be accurate from the beginning. Therefore, all parameters normally set by the execution of a strategy should be set at initialization. First, a snap shot is taken of the state of the system in the connected mode. Thereafter, the driver agents are initialized with the right value to their parameters.

CONNECTED The sub question is: how is the connected mode of the tool which can predict the future locations of the trains implemented?

In this mode agents do not yet exist. It is the rolling stock which is connected to the live feed in the connected mode. It should be seen as if the rolling stock is controlled by the actual drivers.

The rolling stock is connected to the live-feed via the following methods. A method to process pass joint events and a method to process change plan rules events. These methods correct the location and plan rules of the rolling stock. Furthermore, the relevant sections are reserved for the rolling stock. If necessary the rolling stock is deleted.

10.2 EVALUATION OF THE PROGRAM

The program can be evaluated in two ways. The functionality of the program has to be evaluated in order to check whether the hypothesized answer to the sub question about the simplest model possible was correct. This evaluation of its functionality is explained in the following paragraphs. Furthermore, the correctness of the code of the program has to be evaluated. The code is constantly tested during the development of the program. No errors occur and exceptional situations are tested. However, it is not discussed here since this program is just a prototype of the actual tool. It is more important to the development of the actual tool to focus on the functionality of this program then the code itself. This will be done in the following paragraphs.

The performance of the connected mode is tested during the development and implementation of the program. This is not further discussed here. The tests focuses on the performance of the disconnected mode.

The first paragraph explains what aspects of the implemented model need to be evaluated when testing the performance of the tool. The second paragraph explains how the tests are conducted and how the data should be evaluated. Before discussing the results in the last paragraphs of this section, it is briefly explained how the charts(Figure 26, Figure 27 and Figure 28), which show the results, are to be interpreted.

WHAT IS TESTED The local traffic controllers mostly use the safety viewpoint to indicate the location of a train. Therefore, they are primarily interested in knowing the section in which each train is located. Because of this, the most important thing is that in the result of a simulation run all trains are located in the correct section. Hence, the disconnected mode should be compared to the connected mode in the following way: for each section it should be checked whether it is, at the same time, occupied or reserved by the same rolling stock or whether the section is free in both modes.

The following data should therefore be recorded during the time the disconnected mode is activated in a run of the program: in a test run the disconnected mode is activated for five minutes. At each second while the disconnected mode is activated it should be recorded for both the connected mode and the disconnected mode which sections were occupied, reserved and free. Furthermore, at each second it has to be checked how many sections are occupied or reserved for the same rolling stock in both modes or are free in both modes. It is interesting to see the evolution of the performance of the tool through time, since this gives an indication of the size of the time horizon of the prediction. Since a longer time horizon brings with it the risk of producing a less solid outcome due to dispatching strategies during the simulation, in this first iteration a time horizon of five minutes is chosen.

Furthermore, the assumption underlying this implementation, that the focus on a specific yard of the infrastructure of the railway system provides no problem to the performance of the tool, should be tested. This will show whether it is necessary to take the entire infrastructure of the railway system into account or whether a specific yard may suffice.

Finally, it should be tested whether more complex drive behavior of the agents leads to better performance of the tool. For example, the complexity of the drive behavior might lay in the fact that some drivers might accelerate slowly and others as fast as possible.

HOW THE TOOL IS TESTED The performance of the disconnected mode is tested with a data set which is recorded in order to make sure the specific data itself is not of influence to the tests. Furthermore, each test is conducted three times in which the disconnected mode is activated at different moments after the connected mode is started. The disconnected mode is activated 15, 30 and 60 minutes after the connected mode is started.

During the test runs both the disconnected and the connected mode handle time at normal speed. At every second in which the disconnected mode was activated for each section in both modes, it was recorded whether the section was free, occupied or reserved for some rolling stock. Furthermore, it is recorded whether the section is free in both modes or occupied or reserved by the same rolling stock in both modes. It is thereby assumed that the more overlap between the two modes, the better the performance of the tool is.

Three different implementations of the model are tested in which only one variable is changed. This makes it possible to compare the test results of the different implementations. First of all the implementation of the model as described in the previous chapters is tested as described above. This implementation includes the addition of rolling stock during the disconnected mode due to the focus on yard Amersfoort.

Secondly, an implementation without addition of rolling stock during the disconnected mode is tested in order to test whether the assumption that a smaller focus area is of no problem to the performance of the tool. It will furthermore indicate whether a solution should be found for the borders of the railway infrastructure.

Finally, a test is conducted in which natural laws do not play part in the changing of speed of a rolling stock. If the results of this test are compared to the results of the test of the first implementation it can be stated whether a model in which complex drive behavior is taken into account might lead to better performance of the tool.

The conclusions which will be derived from the comparison of the outcome of the different tests will form the basis of the recommendations offered for further research and development of the tool.

HOW TO READ THE CHARTS The results of the different tests can be found in the charts in Figure 26, Figure 27 and Figure 28. On the X-axis the time is represented as the amount of seconds that have passed since the connected mode started. So each column of charts in each figure show the recorded data during the five minutes of activation of the disconnected mode in each test run. The most left column shows the activation after 15 minutes, the middle column shows the activation after 30 minutes and the right column shows the activation after 60 minutes. On the Y-axis the number of sections is represented. Figure 26 shows the test results of the three tests conducted on the implementation in which the natural laws apply and rolling stock is added during the connected mode. From now on this implementation is called the default implementation. Figure 27 shows the test results of the three tests conducted on the implementation in which the natural laws apply, but where rolling stock is not added during the connected mode. This implementation is referred to as the implementation without additional rolling stock. Figure 28 shows the test results of the three tests conducted on the implementation in which the natural laws are not taken into account but rolling stock is added during the connected mode. It is referred to as the implementation without natural laws.

Firstly, the default implementation is discussed. Secondly, the implementation without additional rolling stock is compared to the default implementation. Lastly, the implementation without natural laws is compared to the default implementation.

RESULTS OF DEFAULT IMPLEMENTATION The charts in Figure 26 show that the agents do not occur at the same locations in the disconnected mode as they do in the connected mode at the same time. This can be observed in the upper row of charts in which the amount of occupied sections in both modes is shown and the amount of sections occupied by the same rolling stock at the same moment in both modes. Furthermore, the reserved sections are quite different as well. This is visualized in the second row of charts which shows the amount of reserved sections in both modes and shows the amount of sections reserved by the same rolling stock at the same moment in both modes. The third line shows the amount of free sections in both modes and the amount of sections which are free in both modes. This is obviously not correct either. This is a direct result of the differences in occupied and reserved sections in the both modes. The possible causes for these differences in occupation and reservation are discussed below. First the occupation is discussed, followed by a discussion of the reservations.

The charts in Figure 26 show that in two of the three test runs there are at the start of the disconnected mode as many sections occupied in the disconnected mode as in the connected mode. This means that all active rolling stock in the connected mode is copied into the disconnected mode and driver agents are initialized for all rolling stocks during the transition phase between the two modes. The reason for the fact that this is not the case in the run which started after 60 minutes can have two causes. The first possible cause is that the rolling stock was already at the end of its route. If this is the case, the rolling stock got eliminated correctly. The second possible cause is that no plan rules were known for this rolling stock. In case many rolling stock drove over the railway infrastructure without plan rules this might become a problem for the tool. In this implementation a plan rule is necessary to determine the route of a rolling stock. Therefore, the only way to handle rolling stock without plan rule is to eliminate it from the simulation. If the latter cause is the case a solution should be found, since in that case many sections would be set free illegally (i.e. in the connected mode the section is still occupied) which might influence the simulation. A solution for this problem is proposed in the section *recommendations*.

Furthermore, it can be observed from the charts in Figure 26 that from the start not the same sections are occupied. Rather than a fault in the program, this is probably a result of the errors in the determination of the exact location and the speed in the connected mode.

This is due to the lack of correct data on the location of joints on the infrastructure. Joints might not be in the right place, since it is assumed that joints are equally divided over the railway tracks. The speed of the rolling stock is determined by the location of the joints in the implementation. The speed determines the location on the edge. Therefore the location of the rolling stock might already be at the end of the edge in the connected mode at the moment the disconnected mode is activated, which results in a transition to the next edge at the very start of the simulation run. However, in the connected mode the rolling stock will still be at the end of the edge until a new joint pass event for that rolling stock arrives. These problems would be solved if the data about locations of all the joints of sections would be more accurate. In that case, a more accurate speed could be calculated in the connected mode which lead to a more accurate calculation of the actual location of the rolling stock.

The reserved sections vary a lot in the two modes as well. The reservation of sections depends on the speed of the rolling stock. This speed might be very different in both modes, which results in a difference in reserved sections. This probably has the same reasons as discussed above. A more accurate location of the joints of each section should be known. In addition, the actual maximum speed of each section should be known as well as this might also affect the location of the rolling stock and the reservation of sections.

Besides these problems it can be carefully stated that the techniques to connect rolling stock to the live-feed and to initialize agents at the start of the disconnected mode functions in general. However, the date which should be collected from the live-feed, and the data used to built the environment should be more accurate.

NO ADDITIONAL ROLLING STOCK In this implementation the methods to add rolling stock during the disconnected mode are eliminated to test whether the program should also work if only a small part of the railway infrastructure is taken into account. If the performance of the tool decreases this test shows that it is important to take the whole infrastructure and all rolling stock on it into account. Furthermore, it shows that some solution should be found for rolling stock which starts it journey while the disconnected mode is already activated. Figure 27 shows the results of these test. The first row of charts show that the amount of active rolling stocks decreases fast in the disconnected mode, which is not the case in the connected mode. Compared to the first row of charts in Figure 26 the performance of this implementation of the tool is much worse. The middle row also shows a performance which is worse in the implementation without adding new rolling stock in the disconnected mode than the implementation in which rolling stock is added in the disconnected mode.

From this it can be concluded that a focus on a part of the railway infrastructure is not to be preferred. It is important to take the entire railway system into account. Furthermore, this result shows that it is important to find a solution to inject rolling stock which will start its journey after the disconnected mode is activated in the simulation. A solution might be to have these rolling stock activated at their scheduled time of departure.

NO NATURAL LAWS The purpose of the last test is to show the influence of the natural laws on the state of the railway system. If rolling stock can change from any velocity to any other velocity in zero time, reservations are not necessary as there is no after-running path. This can indeed be observed in the second row of Figure 28. This of course results in a slight increase of the amount of overlapping free sections since no sections are reserved in both modes. Thus, the only difference lies in occupied sections.

In the upper row of charts in Figure 28 the occupied sections in both modes and the overlap is visualized. If this is compared to the upper row of the charts in Figure 26 it can be concluded that the performance of the tool is better in the implementation with natural laws. Both in number of rolling stock active in the disconnected mode as well as with equally occupied sections. From this it can be concluded that drive behavior which is more complex is beneficial for the performance of the tool.

10.3 ANSWER TO THE RESEARCH QUESTION

The question to answer is: what aspects of the state of the trains in the Dutch railway network and aspects of the network itself are relevant and what technologies are necessary to make an agent-based simulation tool which is able to correctly simulate railway scenarios which predict (based on the current location of the trains, other relevant aspects of the state of the trains and the railway network) the future locations of the trains in the Dutch railway network?

From the evaluation of the implementation made for this research project it became clear that it is necessary to have an environment in which the location of the joints and switches are correctly set. It is furthermore necessary that this environment contains the complete infrastructure of the Dutch railway system.

From the sub questions and the development of the prototype of the tool the following became clear. Firstly, it is best to develop two modes in which the rolling stock can be controlled in the environment.

Secondly, no agents exist in the connected mode. The rolling stocks are controlled by the actual drivers from the real system via events pushed by systems of ProRail. It is important that these events are pushed real time and contain correct information. The minimum data necessary from these events is the joint which is passed at that time, the journey number and the plan rules which are actually set for that journey number. This information has to be up to date at any time in order to be able to start a simulation at any point.

Thirdly, at the start of a simulation the agents need to be initialized with all parameters, which are normally set by execution of strategies, set correctly.

The evaluation of the first iteration of the model process furthermore showed that the performance of the tool increases if the drive behavior of the agents and the rolling stock correctly represents the behavior of actual drivers and actual rolling stock.

10.4 RECOMMENDATIONS FOR FURTHER RESEARCH AND DEVELOPMENT OF THE TOOL

The answers to the sub questions and the answer to the main question show that it is possible to develop a tool which can support the local traffic controllers in predicting the future locations of trains in the railway system. A clear description of what data and techniques are necessary to be able to make a transition from the connected mode to the disconnected mode is given. The rolling stock should be controlled by the actual system in the connected mode and by driver agents in the disconnected mode. The parameters of the driver agents should be set to the right value at the start of the disconnected mode. It is possible to use the real world as the start input of the simulation of railway scenarios. The most important benefits of an agent-based approach are shown: the problem can be solved in a natural way, it is rather simple to change or expand the behavior of the agents and the results of new scenarios can be predicted.

The tests of the three different implementations of the program have shown the abilities of the tool. Furthermore, it showed several ways to improve the tool which can lead to a better performance. The recommendations for further research and development of the tool are given below. If Ordina continues this project and follows these recommendations a tool can be developed which supports the traffic controllers of ProRail in detecting conflicts in the railway system beforehand.

Before the behavior of the agents are improved, it is important to improve the implementation of the infrastructure as shown by the results of the tests. The environment needs to consist of the complete infrastructure. Furthermore, the locations of all the joints, switches and end nodes should be correct in order to be able to correctly locate the rolling stock and determine the speed. The starting points and end points of all plan rules should be known and a better live-feed, in which less errors occur, should be provided.

To be able to correctly connect the program to the live-feed it is necessary to implement a method which can handle events an old plan rule is overruled by a new one. This method is quite simple. It should add plan rules to the right rolling stock in the right order. Furthermore it should check whether it is a new plan rule or if it replaces an old plan rule.

After implementing these recommendations, the program should be tested again in order to evaluate the model as proposed in Chapter 5. There are two slightly different implementations which might be interesting to compare with the implementation as proposed above. If these low cost and quite simple implementations improve the results this would be beneficial.

The first assumption which should be tested is if the length of a rolling stock is not of relevance to the state of the railway system. The second interesting test would be to check whether it is necessary or not to implement a method which injects new rolling stock controlled by a driver at the locations where new journeys start during the simulation.

After this is done, it is interesting to improve the MAS. Four improvements are recommended. Firstly, it would be interesting to develop a driver agent which has improved drive behavior. Secondly, it will be interesting to include stopping at stop locations and adding the other goal of the drivers; to arrive on time at planned stops. By doing so it might be necessary to implement some sort of organization which overlooks the agents and fines agents if violating a soft constraint. This is a third improvement. By doing so the agents need to deliberate whether it is more important to arrive on time at a certain stop than to be fined. A fourth improvement might be to include local traffic controllers as agents that set the routes of the agents, since this is how it is done in the original system. This could also solve the problem, discussed in the evaluation, of those rolling stocks that have no plan rules. Furthermore it should be taken into account what are possible next nodes after a switch is passed. This should be implemented in the environment.

These improvements will lead to a tool which can be of great use to the railway traffic. It will be able to alarm traffic controllers of conflicts which would have been missed otherwise. This will make the chance of delays smaller, which will consequently lead to less unsatisfied customers, lower economic costs and a better reputation for ProRail and the transport companies such as NS.

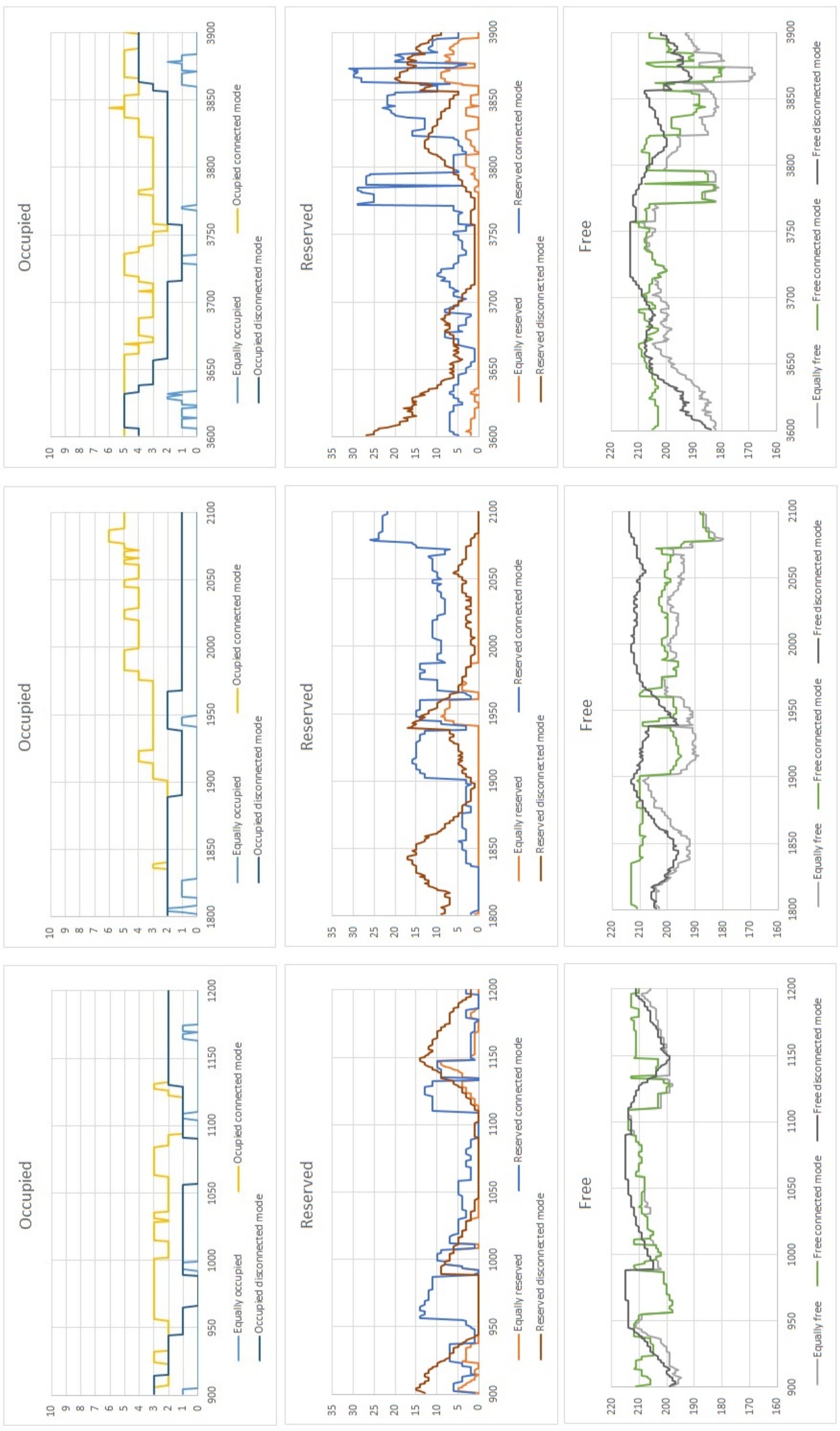


Figure 26: Charts run normal model

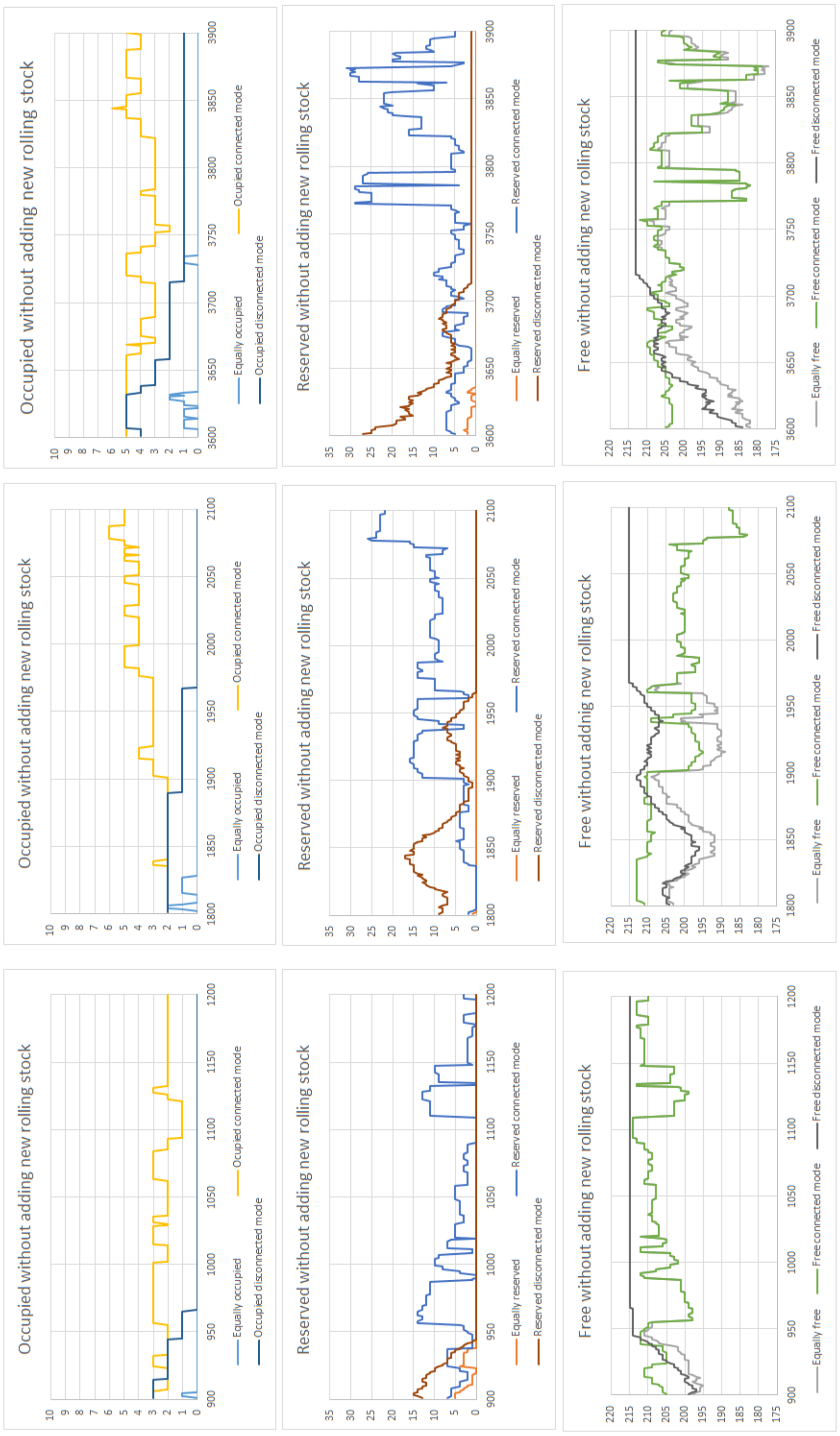


Figure 27: Charts no additional rolling stock

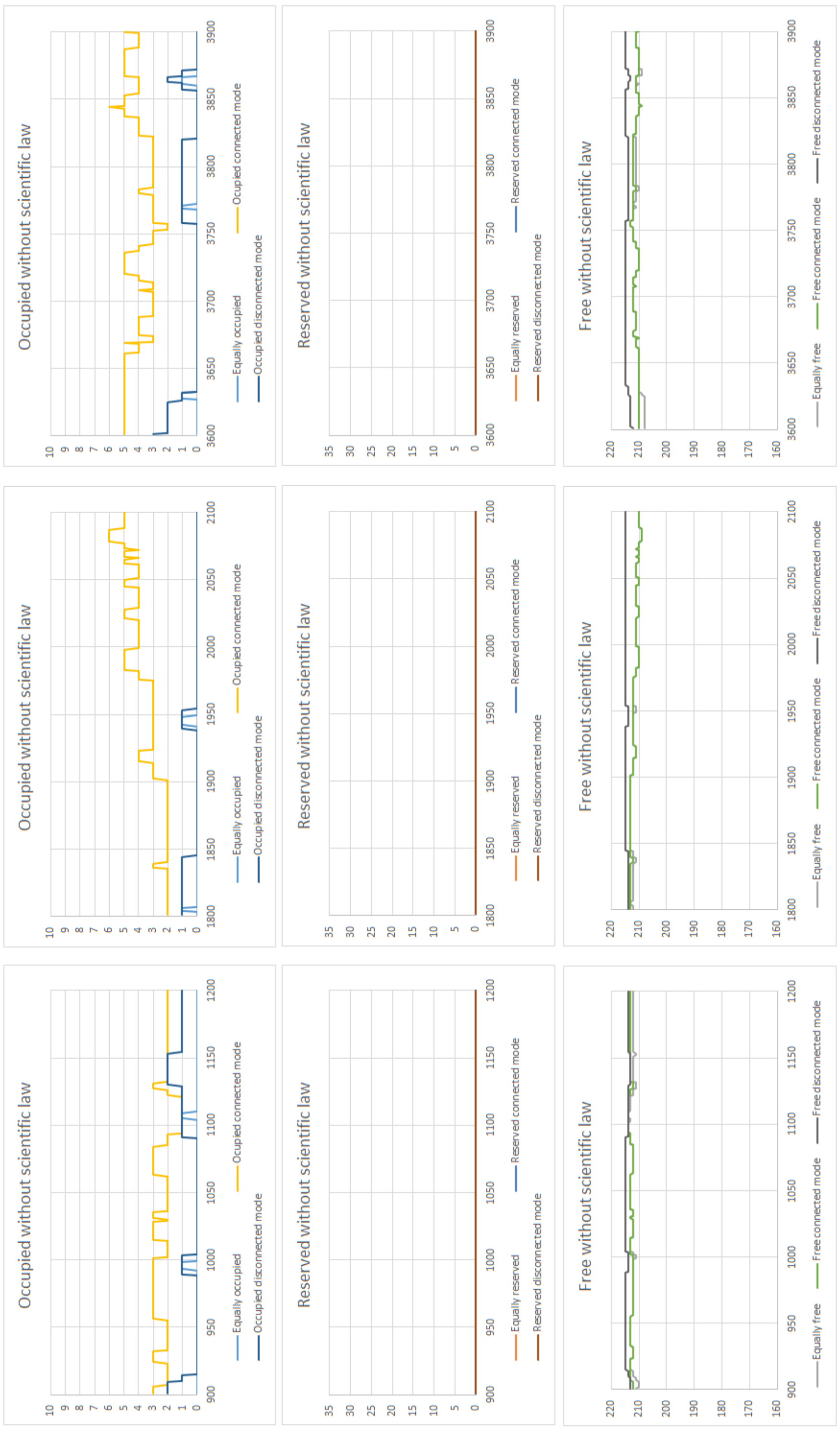


Figure 28: Charts no scientific laws

Part IV

APPENDIX

A INTERVIEWS

A.1 Emiel Sanders, OCCR-ProRail 28-11-2014

Emiel works until the 10th of December 2014 at ProRail. He is a policy-officer for ProRail Incident Management. This division is responsible for the management of railway-incidents. The last few years, Emiel was mainly concerned with the project Spoorweb, which seeks to create a better beeldvorming of incidents.

Objective The objective of this interview was to obtain a better insight in the possible value of the tool and to understand the nature of the problems that ProRail is facing. In addition, I wanted to know more about the way incidents are dealt with within ProRail.

General impression of the interview At first instance, Emiel did not really seem to be convinced of the added value of an agent-based simulation. During the interview, however, he started to see the added value. Nevertheless, Emiel conceived the project as quite ambitious.

My view of the treinverkeersleiding Parts of the things Emiel told me were already familiar to me. My view of the train traffic control thus far corresponded with his information. The division of the levels within the system is indeed, roughly spoken, OCCR, Regional traffic control en local traffic control. Networkprocess management exists as well, but this division is actually part of the OCCR.

Congestion of control points The purpose of OCCR is to prevent control points to congest. This means that there should not arrive more trains at a punt than can be managed at the same time. If else, it turns problematic due to the increasing time of delay. Therefore, the OCCR usually makes sure that trains turn at an earlier point, when they foresee congestion. This method is regulated by protocols as well. So for example, a train that should turn around at Utrecht will now turn around already at Amersfoort. In this way, Utrecht is allayed and the existing delay does not increase.

Infamous control points Some control points are known for congestion when there are certain delays. One of them is Hoofddorp-turning point. This is near control point Schiphol. At Schiphol it is impossible to turn. Therefore, all trains ending at Schiphol continue to Hoofddorp-turning point. Since Hoofddorp-turning point does not dispose of many rails, its limit is easily reached. Due to this small capacity, the delay of 1 train affects all the 3 directions where trains can go to from Hoofddorp-turning point. Therefore, OCCR often choses to turn a delayed train at an earlier place than Hoofddorp-turning point, in order to prevent a snowball-effect of delayed trains. This shows the possible value of the simulation, as it would be interesting to know in advance at what point trains should turn. At present, ProRail already tries to predict this, but it turns out to be very complex without having a simulation.

DISPATCHING STRATEGY Dispatching strategies are the strategies executed by OCCR when they adjust the schedule. This happens about 3000 to 4000 times a day. Usually, this is done on the basis of existing protocols. These can often be adjusted to the specific situation at hand. However, it is rarely known what the consequences of these adjustments are for the rest of the system. It goes without saying that it would be nice to improve this. If it is possible to connect the information of journeys (and smaller journeynumber), conflicts and velocity, much better predictions could be made. Potentially, it becomes possible to answer the question: what consequences does a decision of tdl Amersfoort relating to train x have for train y at control point Hilversum?

PERCENTAGE OF TRAINS ON TIME ProRail generally registers the number of trains arriving in time at their control points; only those at the larger stations. It would be interesting to expand this to more stations in order to have a better overview where measures should be taken to ensure that the minimum thresholds are met.

PROTOCOLS FOR LOCAL TRAFFIC CONTROLLERS Local traffic controllers normally just follow protocols, i.e. if then statements. Such protocols dictate, e.g., that when train 5606 arrive between +0 and +3, train 5607 waits for 5604, only if 5606 arrives with >+3, train 5606 waits for 5607. More experienced treindienstleiders do however sometimes deviate because they know that following the protocol will result in more delays instead of less at places ahead of the train.

INTERESTING INFORMATION FROM SIMULATION

- For the OCCR and the regional traffic controllers: to know a certain minutes in advance whether a control point will become occupied;
- To know from every control point the percentage of trains arriving in time at t+x, given the present infrastructure;

- It would be interesting to be able to see the consequences of an adjustment in the schedule somewhere in the railway system, not only for trains at the particular place where the adjustment is operated, but also for trains at other locations in the system;

TERMEN EN ANDERE FEITJES

- Dispatching stragegy: the OCCR adjust the schedule
- Journey: a certain traject.
- Local traffic controllers do now concentrate between the actual time and 15 minutes ahead. This goes fairly well, but is restricted in their own area.
- Utrecht is most vulnerable to conflicts, Amsterdam is second and Rotterdam third.

A.2 TRAINING REGIONAL TRAFFIC CONTROLLERS - PRORAIL 16-12-2014

Harrie de Haas is trainer at ProRail. He trains people to become regional traffic controller. Together with Gerard, I joined half a day of one of the last training courses of one of his groups. All participants were local traffic controllers. They trained two real-life scenarios with a simulation, thereby explaining to me the purpose and actions of their exercises. I asked some general questions and we visited the training site for local traffic controllers. Moreover, we were given a tour around OCCR and in particular we looked at the national traffic control.

OBJECTIVE The objective was to gain a better insight in the activities and methods of the local traffic controllers and the regional traffic controllers and the systems they use. I also wanted to hear from them what they might consider as additional value of the simulation

GENERAL IMPRESSION OF THE VISIT The visit certainly brought me more insight. Some unclarified issues got clarified and it stroke me again how complex the matter is. I figured that the simulation needed to abstract many aspects, but at the same time several possibilities to do so became clear.

RELATIONSHIP BETWEEN LOCAL, REGIONAL AND NATIONAL TRAFFIC CONTROLLERS The main purpose of local traffic controller is safety. Especially given the strict rules, little initiative is required. There are 13 traffic local control posts, all with their own regional traffic controller. In addition, there is a national traffic controller at the OCCR (this function is normally divided between two persons and one stage manager). When there is a disturbance at his yard, the local traffic controller notifies its regional traffic controller. It is up to the regional traffic controller to decide which infra available is. If the procedure is followed, the regional traffic controller has informed the backoffice (part of OCCR) already at that moment. The regional traffic controller will seek to prevent the delay to increase, by canceling trains and make the right announcements. If necessary he will also inform other regional traffic controller. Meanwhile, the national traffic controller checks the regional traffic controller and verifies if the taken measures do not cause other problems elsewhere. The national traffic controller is also entrusted with the redirection of freight transport.

LOCAL TRAFFIC CONTROLLER The local traffic controller is primarily concerned with the safety at the railways. He finds himself supported by multiple safety systems, but he has not much powers. It seemed that not following the protocols can cause him serious reprimands. At the other hand, regional traffic controllers are often of the opinion that the amount of delays really depend upon the specific person that is local traffic controller. Therefore, this aspect will need further abstraction for the simulation. Local traffic controller determine the specific rails, hence this is the way a TDL can influence the arrival time of trains.

The information at the local traffic controller's disposal is presented at several screens. The TDL can also call machinists. In practice, this goes wrong sometimes as machinists call the wrong numbers. The local traffic controller can ask its regional traffic controller for help. The information presented at the screens exists of an actual timetable, maps of the yard showing the trains and position of switches and signals and a time/place diagram that shows which train will arrive at which platform. Gerard knew from experience that this latter screen is not always used. The timetable exists of plan rules. These show the departing and arriving platforms of a train, the time of departure and the time that will be set by ARI (automatische rijweg instelling or automatic route setting). The time set by ARI can be adjusted and this is the reason why signals en switches can be set at a later moment. A local traffic controller has the capacity to manually set a route. At all times, however, the safety systems remain active. Therefore, if a conflicting is still set, it will not apply to the entire route.

The maps of the different yards relevant for a local traffic controller show which sections are occupied. The local traffic controller cannot see why these sections are occupied. It might imply a driving train, but other reasons are possible as well, such as a defect in the safety system. A more detailed map shows the color of the signal, the position of the switch and what route is set. So the important thing to realize is that a local traffic controller can merely see which sections are occupied, without knowing the precise location of a train.

Due to the time/place diagram it is possible to know in advance whether two trains will need the same rails at the same time, or will be problematically close to each other.

In conclusion, although many protocols exist, human experience turns out to be pivotal in the functioning of local traffic controllers.

REGIONAL TRAFFIC CONTROLLERS Whereas local traffic controllers are preponderantly concerned with safety, regional traffic controllers main occupation is the logistic. Being notified by a local traffic controller, the regional traffic controller will take care of a change in the optional infrastructure which is send to the transport companies so they can set a new schedule. Meanwhile, the regional traffic controller will remove planned trains that can no longer continue their journey, either due to the impossibility to continue or due to such a delay that continuing makes no sense anymore. Thus, the main task of the regional traffic controller is to reduce the consequences of a disturbance as much as possible. In its task, the regional traffic controller is supported by the national traffic controller. As mentioned before, the national traffic controller redirects the freight traffic and checks the regional traffic controller and verifies if the taken measures do not cause other problems elsewhere.

A regional traffic controller uses time/place diagram which shows the scheduled trains and their current location and makes changes if necessary. These changes will pop up at the local traffic controller. The functioning of the regional traffic controller is also importantly based upon experience. He uses its screens to check whether the minimum distance between trains is respected, since this might otherwise lead to delays. Its knowledge of the route is essential, as it makes, e.g., quite a difference to know if passes will be disturbed by stopped trains and the amount of free tracks available. It seems to be legitimate to presuppose that regional traffic controllers have a profound knowledge of their area, enabling them to work efficient and accurate.

BIBLIOGRAPHY

[1] Representation of a multi-agent system. http://www.codeproject.com/Articles/13544/Agents-and-Multi-agent-Systems. Accessed: 12-06-2015.

[2] Trail, research school for transport, infrastructure and logistics. http://rstrail.nl/new/home/home-2/. Accessed: 11-05-2015.

[3] Prorail, view. https://www.prorail.nl/vervoerders/diensten-voor-vervoerders/informatiediensten/view. Accessed: 01-07-2015.

[4] Ana LC Bazzan and Franziska Klügl. A review on agent-based technology for traffic and transportation. *The Knowledge Engineering Review*, 29(03):375–403, 2014.

[5] Claus Biederbick and Leena Suhl. Improving the quality of railway dispatching task via agent-based simulation. *Publication of: WIT Press*, 2004.

[6] Bo Chen and Harry H Cheng. A review of the applications of agent technology in traffic and transportation systems. *Intelligent Transportation Systems, IEEE Transactions on*, 11(2):485–497, 2010.

[7] Andrea D'Ariano. *Improving real-time train dispatching: models, algorithms and applications*. Number T2008/6. Netherlands TRAIL Research School, 2008.

[8] Mehdi Dastani and Bas Testerink. From multi-agent programming to object oriented design patterns. In *Engineering Multi-Agent Systems*, pages 204–226. Springer, 2014.

[9] Harrie de Haas and students. Prorail, training regional traffic controllers. Interview, december 16 2014.

[10] Jose Luis Espinosa-Aranda and Ricardo García-Ródenas. A discrete event-based simulation model for real-time traffic management in railways. *Journal of Intelligent Transportation Systems*, 16 (2):94–107, 2012.

[11] P Kecman. *Models for Predictive Railway Traffic Management*. TU Delft, Delft University of Technology, 2014.

[12] Julia Lo, Jop Van den Hoogen, and Sebastiaan Meijer. Using gaming simulation experiments to test railway innovations: implications for validity. In *Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World*, pages 1766–1777. IEEE Press, 2013.

[13] ProRail. ProRail jaarverslag 2013. http://www.jaarverslagprorail2013.nl/jaarverslag-2013/kerncijfers/a1297_Kerncijfers, 2013.

[14] ProRail. Network statement 2015, combined network. Network Statement 3487859, ProRail, 2014.

[15] Emiel Sanders. Prorail, incidentregie. Interview, november 28 2014.

[16] Robert Siegfried. *Modeling and Simulation of Complex Systems: A Framework for Efficient Agent-Based Modeling and Simulation*. Springer, 2014.

[17] Afdeling Transport en Planning Technische Universiteit Delft, faculteit Civiele Techniek en Geowetenschappen. Parlementair onderzoek onderhoud en innovatie spoor. Parliamentary research document 32707 nr. 11, Tweede kamer der staten generaal, 2012.

[18] Roeland van Beek. A statistical model of traffic control. Master thesis, Utrecht University, 2013.

[19] Michael Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, 2009.

[20] Faramak Zandi and Madjid Tavana. An optimal investment scheduling framework for intelligent transportation systems architecture. *Journal of Intelligent Transportation Systems*, 15(3):115–132, 2011.