From log files to train traffic reports: Using Natural Language Generation to explain anomalies from Train Control System log files

Thesis report

Bojana Urumovska

Technische Universiteit Delft

From log files to train traffic reports: Using Natural Language Generation to explain anomalies from Train Control System log files

Thesis report

by

Bojana Urumovska

in partial fulfillment of the requirements for a degree of

Master of Science
in Computer Science
Data Science and Technology Track

at Delft University of Technology

Student number:	4633334
Project duration:	Sep 1, 2018 – August 22, 2019
Supervisor:	Dr. N. Tintarev
Thesis committee:	Prof. dr. ir. G. J. Houben
	Dr. Nava Tintarev
	Dr. Matthijs Spaan

Abstract

The Natural Language Generation field has advanced in generating human readable reports for domain experts in various fields. Nevertheless, Natural Language Generation and anomaly detection techniques have not been used in the rail domain yet. Currently, data analysis and incident reporting for log files from the train control system are performed manually which is very time consuming task that is prone to missing crucial information. The rail domain is safety critical domain where detailed analysis of the train control system may prevent incidents from happening as well as help improve the performance of the train control system. This research designs, implements and evaluates a Natural Language Generation model that successfully translates anomalies detected in log files into human readable reports.

This thesis presents the steps taken for developing a Natural Language Generation system in the rail domain. Additionally, we examine two representations of the train control system used for the Content Determination task of the Natural Language Generation system. Through a case study with domain experts, we evaluate the performance and preference between the reports generated based on the two representations of the train control system and the data retrieved from the log files. The goal is to find a representation that presents the used with a full/solid understanding of the anomalies detected in the log files.

Based on the case study performed to evaluate the system, we present the finding that when developing a Natural Language Generation system for the rail domain, reports generated using a more detailed representation of the train control system (more precisely, using both state names and state attributes that specify the step by step process of setting a route for a train) were preferred over the reports generated using a less detailed representation (only state names). The preference was based on readability, accuracy and understandability measures of the reports presented during the case study.

Keywords Natural Language Generation, Anomaly Detection, Log Files, Rail Domain

Contents

List of Figures

List of Tables

1

Introduction

1.1. Motivation

Natural Language Generation has been used for producing human readable summaries for domain experts to simplify and improve complex tasks that would otherwise require a lot of time and domain knowledge to be performed. Such example are the different BabyTalk[19], [11], [32] Natural Generation systems that have been developed to help medical professionals to do shift-handovers and continuously follow the medical health of babies in neonatal care. The advantages of having this Natural Language Generation systems especially in safety critical environments is the precision the system offers as well as quickly generated human understandable reports based on complex data analysis. Natural Language Generations systems help automatize generating reports from a large amounts of heterogeneous or unstructured data.

Even though the Natural Language Generation field has been advancing and it has been used in variety of domains, it has not been yet implemented in the railway domain. Analyzing the large amounts of data that is being used and produced on daily bases from the train control system required a lot of domain knowledge and due to the nature of the data, it is very time consuming. Rail domain experts have been doing log analysis and writing incidents reports based on manual analysis. As mentioned before, due to the large amount of data available, it is fairly easy for the human eye to miss crucial information in the log files.

This research has a motivation to automatize the generation of incident reports in the rail domain. Additionally, we aim to help prevent incidents by providing detailed human readable reports to the train control system developers about any abnormalities detected in the log files.

As Natural Language Generation and anomaly detection techniques have not been used in the rail domain, one of our biggest challenges is to establish an accurate representation of the train control system that would lead to a solid explanation of the anomalies detected.

Lastly, this research is done in collaboration with CGI B.V Netherlands, who proposed the research topic and requested a solution that would provide human readable reports from the log files of the train control system.

1.2. Research Question and Steps

As stated in the motivation section above, the goal of this research is to design and implement a Natural Language Generation system that would generate human readable summaries that explain anomalies detected in log files from a train control system. We design two representation of the train control system that are used for the first task of the Natural Language system. We examine how well each representation performs and we answer the following research question: When reporting anomalies of a train control system, what degree of detail from the log files should the NLG system use such that

the user gets a full understanding of the anomaly?

To answer the research question we perform a case study and test the following two hypothesis:

- H1: The reports generated by the NLG system that uses the more detailed representation of the train control system is more understandable compared to the reports generated by NLG system that uses the less detailed representation of the train control system
- H2: The reports generated by the NLG system that uses the more detailed representation of the train control system is preferred by the domain experts compared to the reports generated by the NLG system that uses the less detailed representation of the train control system

The research steps taken in this thesis are shown in Figure 1.1

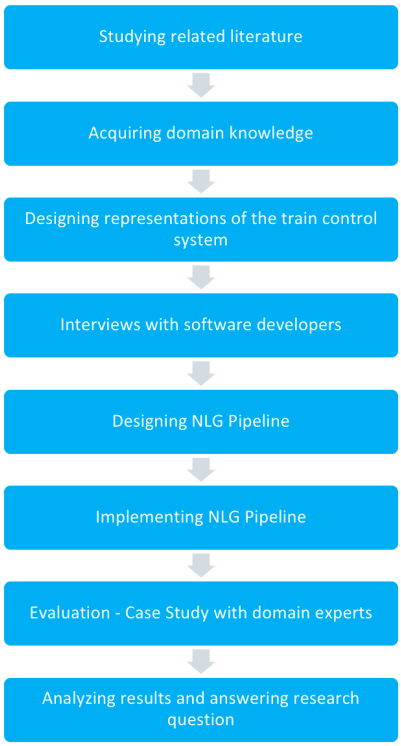

Figure 1.1: Step by step representation of the research work

1.3. Contributions

The contribution of this research is a Natural Language Generation model that generates human readable reports that obtain high performance score when presented to domain experts. Furthermore, the research examined and concluded that using more detailed representation of the train control system for the Content Determination task contributes to generating reports that are preferred over reports that are generated based on a less detailed representation of the train control system. More precisely, extracting and using both the state names and state attributes for the log files helps in generating reports that better explains the anomalies detected. This research shows the benefits of using Natural Language Generation systems in the rail domain.

1.4. Thesis Outline

The structure of the thesis report is as follows:

- **Chapter 2 : Related Work**
  In this chapter the reader is presented with the existing research for the problem in question. The

techniques used to solve related problems are review and explained. Furthermore, the techniques on who to evaluate the methods used are discussed. Lastly, the research gaps and the research question are presented.

- **Chapter 3 : Project Context**
  The Project Context chapter gives a detailed explanation and a general overview of the problem in question. It introduces the train control system that is being analyzed and explains the information that can be found in the log files used.
- **Chapter 4 : Methodology**
  The methodology chapter explains all the techniques used and the steps taken to develop the Natural Language Generation system.
- **Chapter 5 : Evaluation**
  The evaluation section covers all the techniques used for evaluating the system. It gives a detailed explanation of the case study performed. Furthermore, it presents the reader with the hypothesis of the research, the results from the case study and the answers of the hypothesis tested.
- **Chapter 6 : Discussion and Future Work**
  In this chapter a summary of the research is presented, after which the results are discussed and the research question is answered. Additionally, suggestions for future research are given.

2

Related Work

This literature study presents the reader with the background knowledge on the existing techniques of Natural language generation as well as some background knowledge on graph-based anomaly detection. Based on this literature study the methods used during the research will be chosen.

2.1. Introduction

This chapter presents the state-of-art literature in the field of Natural Language Generation and graph-based anomaly detection. It analyses the fields in which this techniques are used and the methods used for implementing them. In Section 2.2 an introduction of Natural Language Generation is presented as well as an overview of the existing methods used in NLG. Furthermore, section 2.3 presents the challenges one may encounter when developing an NLG system. Then in Section 2.4, some related problems are discussed and compared to the problem of the research. Section 2.5 presents the reader with the techniques used for evaluation Natural Language Generation systems. Section 2.6 presents background knowledge of graph-based anomaly detection. Literature in graph based anomaly detection is presented as anomaly detection is required for performing one of the Natural Language Generation tasks. In Section 2.7 the research gaps and the research question are introduced.

Based on this literature study the methods used during the research will be chosen. The main goal of this chapter is to present the reader with the basic knowledge of the techniques that will be used. Furthermore, to show in which other aspects this techniques can be used and what is the connection between existing research/literature and the research in this thesis work.

2.2. What is Natural Language Generation

Natural Language generation falls within the natural language processing field. It is a method that is used for generating natural language from a given set of text or data. Therefore, two task that a natural language generation system can solve are data-to-text or text-to-text problems. As this thesis is dealing with machine generated data (log files), the task to be performed is data-to-text.

Definition : Data-To-Text Natural Language Generation
Given a non-linguistic input
Generate text or speech in a natural (human readable/understanding) language

Data-to-text NLG is also characterized as ”the sub field of artificial intelligence and computational linguistics that is concerned with the construction of computer systems that can produce understandable text in English or another human language from some underlying non-linguistic representation of information” [22].

When solving an NLG problem there are 6 general tasks that need to be performed:

- *Content Determination* focuses on deciding what data is relevant for the audience and therefore should be communicated
- *Text Structuring* is a step where the order of presenting the data is decided upon (e.g. this can be from temporal order, to importance or any other grouping that the scientist might decide on, based on the communication purpose)
- *Sentence aggregation* is the process by which related text is merged together and put (grouped) in a single sentence. This task is usually domain dependent. One example for this is a football match reporting system. If one player scores 2 or more goals during a match, instead of reporting each goal in a separate sentence, the sentence aggregation step will enable the summary to have one sentence which reports the name of the player and all the goals he scored with a consecutive time of when each of the goals was scored. This task contributes to making the summaries look less machine generated as repetitiveness of same sentences is avoided.
- *Lexicalization* is the task in which the content is fully known and a natural way of expressing the problem needs to be decided upon. Deciding on what would be a natural way to express some content might be rather difficult as in natural language there is a variety of ways to say one same thing. Whether a system should use a consistent way of communicating the information or diversity is important, depends on the domain. Additionally, in this task, some systems may even need to adapt to other considerations such as attitude or affect towards the content that is being communicated. A simple way to perform this task would be to operate on pre-verbal messages, converting domain concepts directly into lexical items.
- *Referring expression generation (REG)* deals with recognizing and communicating different domain entities in a correct way. For example, when an entity is mentioned in the summary for the first time, one needs to decide the way the entity will be introduced. Furthermore, REG takes care that the text is synchronized and once an entity has been mentioned, it will be refereed to in later appearances.
- *Linguistic realization* makes sure that each sentence in the summary is following all morphological rules. In this task one also needs to handle auxiliary words as well as correct punctuation of the sentences. Additionally, for some languages one also needs to take care of correct verb conjugation and similar linguistic rules. There are plenty of methods researchers have came up with for completing this task. The tree main once are: *human crafted templates, human-crafted grammar-based systems and statistical approaches*

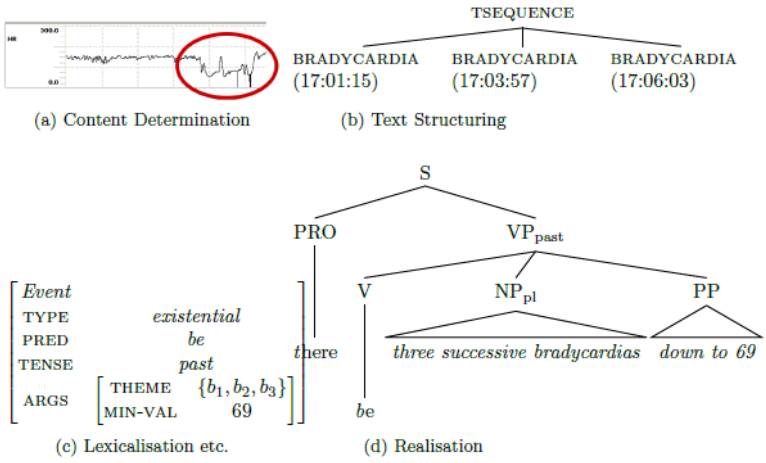

Figure 2.1: Overview of the NLG tasks through an example from [10]

Figure 2.1 shows a overview of the above explained tasks through an example from a NLG system that produces medical reports for babies that are in neonatal intensive care.

Within NLG there are several approaches that one can take which may focus more on different tasks depending on the problem that is being solved. Different NLG architectures may focus on one or more of this tasks. The first and most general NLG architecture was defined by Reiter et al. in [21]. This pipeline is called the modular architecture and it divides the above mentioned tasks into different modules and treats the modules with a very clear division from each other.

The main NLG approaches/architectures are :

- *Modular Architectures* : Modular architectures involve very clear division between the subtasks with a significant variation between them. The most general and standard pipeline for modular architecture in NLG is introduced originally by Reiter and Dale and it has been generated based on actual practice. The pipeline is shown in Figure 2.2. Combination of the above mentioned subtasks are implemented in different modules. For example, the *Text planning* module is mainly concerned with deciding on ”what to say”. In this step the content determination and text structuring tasks are performed. The output of the first module is a text plan. This plan is then given as input to the Sentence planner which performs the sentence aggregation, lexicalization and referring expression generation tasks. This is seen as the ”how to say it” process. The last module is the Realiser which uses the output of the Sentence Planner. It implements the last task which is the linguistic realization task. It generates the final sentences in a grammatically correct way by applying syntactic and morphological rules. This is only one example and the most general use of a modular architecture and therefore different variations exist. For getting informed about the majority of other modular approaches proposed in different NLG literature the reader is refereed to look at the [10].

  Furthermore, a special attention is given to the modified version of general modular NLG system architecture. The newly suggested NLG system is adapted for systems whose input is raw data instead of AI knowledge base. The difference of this (Figure 4.1) architecture to the general one is that this architecture has two additional steps in the pipeline. A Signal analysis accompanied with the data interpretation step. This two steps have been added to the data-to-text system when dealing with raw data as the system must first analyze and interpret the data after which one can decide how to linguistically communicate it. Below the modified pipeline is explained in detailed.

  1. Signal Analysis: Reiter has defined this step as the step where numerical data is being analyzed by looking for trends and patterns in the data. He states that the goal of signal analysis is to replace numerical data by a set of discrete patterns. He furthermore explains that when the data is structured this step can be skipped. An important aspect that one needs to take care of in this step is that in case the data includes any noise, this is the time when the noise need to be removed.
  2. Data Interpretation: This stage interprets the output of the signal analysis. It relates patterns and trends found to the data and infers more domain-specific messages. Furthermore, it decides how important an event is. Lastly, it explores if there is some relationship between events detected.
     For example, if the system is producing medical reports for babies the system analysis outputs 3 events that are important. Assume one shows a decrease in heart rate , another shows stabilization of the heart rate later on. The third event is that some medicine was given to the baby. The Data Interpretation stage can relate this and interpret it as the baby had a low heart rate but after the medicine was given, it normalized. So the normalization of the heart rate is associated with the medicine given. Multiple messages/relationships can be created in this stage, given the patterns and trends found in the data.
  3. Document Planning: Document Planning takes all the messages that were produced by the data interpreter and chooses which of this messages are important and should be communicated to the user. The structure of the text is also constructed in this stage. This task is very important when a lot of messages have been produced by previous step(s) and not all of them should be communicated. On the other hand, if the case is such that all patterns and information provided needs to be communicated, then the document planning step is a very straight forward task.

4. Microplanning and Realisation: Microplanning and Realisation is the last stage that outputs the finalized syntactically and grammatically correct natural language explanation. This stage is a merge of the microplanning and realization stages given in the standard modular architecture presented in Figure 2.2

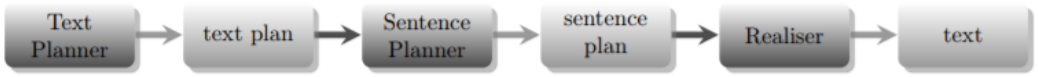

Figure 2.2: Most general modular architecture pipeline for NLG systems [23]

- *Planning based Approaches* : In a plan-based NLG approach there are no rules regarding which actions will form the plan, therefore it is possible to combine elements from strategic (what) and tactical (how) nature. In that way the problem of "what and how" can be approached with a same set of operations. The main concept of this approach is that text generation can be viewed as the execution of planned behavior to achieve a communicative goal, where each action leads to a new state. As seen in literature there are two main ways to design plan based approaches. Planning through Grammar or Stochastic Planning under uncertainty using reinforcement learning.

- *Integrated or global approaches* : The integrated or global approaches became widely used and developed in a later stage of the NLG development. Their rise was mostly because of the growth of data availability, the growth of computational power and the research developments in data analysis. Due to the challenges the modular approaches experiences such as limitation on generalizing a model due to specificity on developing very domain specific solutions, the integrated approaches tent to emphasize the use of statistics in the NLG process. This approaches tend to take a cooperative approach rather then modular. Integrated approaches are a more advanced level of NLG in which the NLG problem may be seen as sequential process, a classification or optimization problem. Current research is being performed on ways to avoid developing domain specific NLG systems. The existing solutions on this are models that used advanced up to date neural network models as well as other machine learning techniques. Moreover, there is research performed where NLG was approached by deep learning methods.

  As this research will follow a modular approach, we do not discuss the integrated and global approaches in details.

2.3. Challenges in NLG

Even though a lot of research has been done in the NLG filed and it is generously advanced, there are still some challenges one might encounter when developing an NLG system.

One major challenge that might arise when developing a Natural Language Generation system that is very domain specific and is developed for experts in a field is the Content Determination part of the NLG. In this cases, the people that develop the system must get a sound understanding of the domain as well. This will prepare the developer/researcher to make a good decision in which information are important and therefore should be presented in the generated text. This is challenging and expensive as it might require a variety of domain specific training for a solid understanding of the domain to be acquired.

Another challenge is to be able to interpret the information. This fall within the text structuring task. Once all the important information have been identified, one needs to know how to interpret and explain them on a level of the expert that will be reading the text. An example of such a system is the BabyTalk project [19]. The BabyTalk project requires a lot of medical knowledge when performing both of the tasks mentioned above. As the goal of this research is to produce summaries from data of a train control system, there is a need on understating the specific software as well as the train traffic domain. Therefore this is one of the first challenges we encounter in this research.

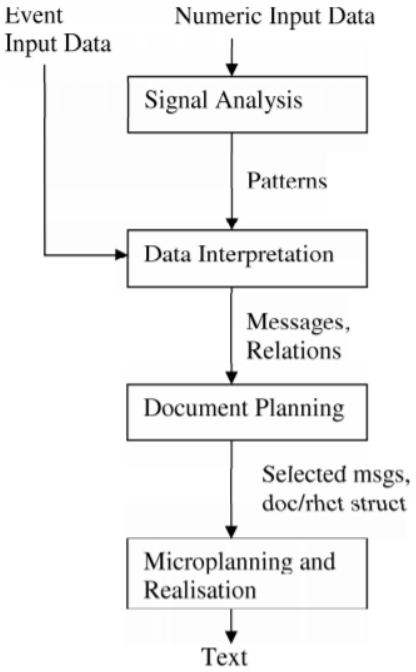

Figure 2.3: Architecture for data-to-text NLG systems defined by Reiter in 2007[23]

Finally, when looking in methods for evaluating the system, one might be limited to the choices as some methods require a golden standard text. This can be a text produced by a human or another NLG system. Unfortunately, if there is not an existing NLG system that can handle the data set or the domain, then there is no golden standard text. Additionally, if there is a lot that needs to be analyzed for a particular report to be produced, then asking a human to do that is not efficient and may be very expensive. This is another challenge we face with this research.

2.4. The usage of NLG systems over the year in different fields

In this section the state of art of NLG is presented. The NLG methods and architectures used for specific problems and the latest developments of the field will be also covered. The domains in which the NLG systems got a lot of attention are discussed in different subsections. Furthermore, domains where NLG is in the early stage of development are discussed in the last subsection.

2.4.1. NLG in medical field

As mentioned in the introduction section, currently, there is a substantial development of NLG systems in the medical field. On of the earliest data-to-text NLG systems - TOPAZ was developed in 1991. It was used for generating summaries from blood cells count and drug intake data of lymphoma patients. The system was a schema based system that helped clinicians in analyzing the data from the patients. This NLG system along with others was developed for discrete numerical data. Moreover, there are also systems that generate summaries for patients rather then medical staff. Later on, [19], contributed to a development of a NLG system that is used in the neonatal intensive care. The project is called BabyTalk and it is implemented such that it can produce different types of reports based on who the target user is. BT-45 generates summaries from clinical data from 45 minutes and it is to help medical staff make decisions. BT-45's input data included both the continuously monitored physiological data and supplemental discrete information which was collected for the purpose on the ward by a research nurse. The BT-Nurse [11] [32] system on the other hand, generates summaries from data gathered for 12 hours and the system is developed to help nurse shift handover. Handover is a very patient safety critical process. The system has been designed to use only data that is available in electronic form that represents the routinely checkups of the infants during a shift. The system contributes to making the handover more safe and easy as it addressed the decision on which data is the most relevant for the

next shift given the high volume of heterogeneous data. Lastly, the BabyTalk project has also been developed to create summaries on data from 24 hours but in a less medical specific language. This summaries are generated for the parents, hence the tool is called BT-Family [13]. Furthermore, there was a intention to also develop a BT-Clan tool, that would generate summaries with the status of the baby and the parents and it was meant to serve information to friends and family of the baby. This tool was intended to include information about the status of the parents as well, but due to lack of data on the status of the families the tool was not developed. The pipelines and evaluation methods used for the above mentioned NLG systems are specified in Table 2.4.3.

From all the above mentioned NLG system, the BT-45 will be discussed in more detailed, due to the similarity of the problem defined in that research and the problem we are working on. One main similarity is the type of reports generated by the system. BT-45 generates reports for experts in the filed, which is also the goal of this research work. Additionally, the BT-45 system is using raw data generated by machines which is the case with the the data used in this research. Due to the nature of the data, in the BT-45 system the modern modular approach pipeline for data-to-text NLG is used. BT-45 system takes raw data (sensor data and event logs) as input. The first task of the BT-45 system is concerned with signal analysis (analyzing the raw data in order to find patterns and trends). The next task performed is the data interpretation task where the patterns found in the signal analysis are interpreted as a medical expert level messages and correctional and or causal relationship between the messages is searched for.

2.4.2. NLG in weather forecasting

The trend on using NLG for weather forecasting started as early as the 1991 by developing weather forecasting systems. The very first systems developed were the [16], [25], [12], [8]. One of the later on developed NLG systems is the SUMTIME-MOUSAM and it is a marine forecast generator that produces textual marine weather forecasts for offshore oilrig applications. The interesting aspect about this system is that forecasters can tailor the output text using control data derived from end user profiles. Furthermore, the summaries generated are post-edited by forecasters before the summaries is hand over to the end users. The data used for generating the summaries is time series data from Numerical Weather Prediction (NWP) models. The dataset includes around 40 weather parameters from a time interval of 3 hours with weather prediction up to 72 hours from the moment the forecast is issued. The SUMTIME-MOUSAM system used the general modular NLG architecture Figure 2.2. Finally, the evaluation of the system is done by measuring the edits forecasters make after the system gives the output and before the summary is given to the end user.

2.4.3. NLG in other fields

The NLG systems have been also developed in other fields such as football reports, scuba diving reports and many others. The NLG system have been developed to be able to handle big amounts of heterogeneous data. Further reading in the above mentioned fields can be found in the following papers.

- Scuba diving reports [27] [28]
- Football match reports [29] [7]

Paper	Year	NLG Architecture used	Goal	Evaluation method used
BT-Nurse, Hunter at al. [11]	2012	Data-to-text pipeline 4.1	Generating 12h medical reports for nurse shift handover	Human Rating
[4]	2013	Modular design 2.2	Producing medical reports from Physiological Data Monitoring	no information
[20]	2009	Modular design 2.2	Generating 45 mins time period medical reports for making fast decisions by medical staff	no information
[13]	2011	Modular design 2.2	Generating 24h medical reports family members of the patient (baby)	no information
[26]	2003	General modular pipeline 2.2	Generating weather forecasting reports	modifying the output and measuring the edits made

2.5. Evaluation of NLG systems

As seen in the previous subsection, NLG is used in a variety of way, most of them safety critical (e.g [19], [27], [11], [20]). NLG systems are developed to perform a rather challenging task and therefore need to be evaluated. The following 3 techniques are being used for evaluating NLG systems: task performance, human rating and metrics (comparison to golden standard). Moreover, the evaluation can be performed under controlled or real world setting. Which method the researcher uses for the evaluation depends on the goal of research or the hypothesis that is being tested.

Below an overview of all the techniques is presented.

- **Task Performance** This evaluation technique measures whether the NLG system achieves its goal. This technique can be performed both in real world or in a laboratory experiment. For example, when one wants to evaluate a behavior change support system that generates text to help an individual develop a habit or get read of a bad habit, the task performance evaluation should measure the success of the NLG text on developing/getting rid of the habit.
- **Human Rating** When using the Human rating evaluation technique, the measures taken to determine the quality of the generated text is *Readability*, *Accuracy* and *Usefulness*. It is an intrinsic evaluation and usually researchers use Likert scale (3,5, or 7 point) in order to gather quantitative feedback. Additionally, researchers sometimes also ask for textual feedback and analyze that feedback as well.
- **Metric** For a metric based evaluation creating a gold standard is the first and most important task. A gold standard (reference text) is a text that is the desired output of the system and it is written based on the same input data that is given to the NLG system that is being evaluated. Usually the text is human-written and it is required to acquire multiple texts from different people. Furthermore, a gold-standard can be generated by an already existing NLG system if one is available. After the gold standard is set, the output of the NLG system (candidate text) and the gold standard text are compared. This can be done using different metrics. Some common metrics for NLG evaluation are BLEU, ROUGE and METEOR and the difference between them is the scoring functions used.
  - *BLEU* is a precision based metric which calculates a score of a generated text based on comparing the generated text with a gold standard text. The BLEU metric can use different length of n-grams for calculating the score of the text. The score is calculated by the formula given below:
    $$BLEU = \frac{m}{n}$$
    where m is the number of n-grams that appeared in both the candidate text and the reference text and n is the total number of words in the candidate text. The output of the BLEU metric is always a number between 1 and 0. Scoring a 1 corresponds to perfect matching of the texts compared.
  - On the other hand, *ROUGE* is a recall based measure. It checks how many n-grams from the reference text appears in the system candidate text. This metric is seen as a weaker one compared to the BLEU metric, as humans can be inconsistent and in the ROUGE metric that might point to not so poor results.
    The ROUGE metric has multiple ways of scoring the text. One way of calculating the ROUGE score is by using a fixed length of n-grams, this technique is called ROUGE-N. The second option is to look for the longest common sub sequence of word between the gold standards and the generated text, this approach is called ROUGE-L. Lastly, one can also generate a ROUGE-S/SU score by calculating the number of skipped bi-grams (pair of words in their sentence order, but allowing for any number of other words to appear between the pair.)
  - Lastly, *METEOR* is a calculated by getting the harmonic mean of recall and precision on unigrams. An additional feature that the METEOR metric has is that it also matches words by looking for synonyms and roots of words (stemming). The METEOR metric gives a higher weight to the recall. The formulas for calculating recall, precision and finally METEOR are the following:
    $$Precision = \frac{m}{w_t}, Recall = \frac{m}{w_r}$$

where m is the number of unigrams in the candidate translation that are also found in the reference translation, w_t is the number of unigrams in the candidate translation and w_r is the number of unigrams in the reference translation.

$$METEOR = F_{mean} = \frac{10PR}{R + 9P}$$

2.6. Graph Based Anomaly detection

This section presents the literature studied that is related to Graph-based anomaly detection. It gives an overview of the methods used in the field and the challenges associated with it. Graph based anomaly detection is part of this research as it will be for the Signal Analysis task which was explained in the Section 2.2. The goal of the NLG system is to explain anomalies found in the log files. In order to be able to achieve this goal, the anomalies need to be detected and this is done by using graph based anomaly detection.

2.6.1. What is graph-based anomaly detection?

Anomaly detection is a field in data-mining and it is identification of data points, items, observations or events that do not conform to the expected pattern of a given group. This means spotting unusual behavior of data points. Anomaly occurs very infrequently but can point out very significant information. Graph based anomaly detection is anomaly detection performed on graph based data. Graph based anomaly detection is vital because of the following reasons:

- The first and main reason for using graph based anomaly is because of the type of anomaly this research is trying to detect. The train control system uses a specific pattern of states when setting a route. This pattern is presented as a graph. As the anomaly to detect in this research is when the stages go in an unusual pattern, the graph-based anomaly detection is chosen
- Another reason for using graph based anomaly detection is the inter-dependent nature of data and the powerful representation graphs provide for this type of problems. [2]. As the data objects are related to each other and exhibit dependencies, a good way to visually show this is by using graphs. Graphs naturally present the inter-dependencies by introducing edges between the related objects. The multiple paths lying between these related objects effectively capture their long-range correlation. Moreover, a graph representation facilitates the representation of rich data sets enabling the incorporation of node and edge attributes/types.

Types of graph-based anomaly detection

As presented by two graph based anomaly detection surveys by Leman at al. [2] and Debajit and Samat [24] graph-based anomaly detection is divided in 4 main parts. *Static vs. Dynamic* graph anomaly detection and *Plain vs. Attributed*.

Anomaly detection on *static graphs* is performed in order to spot anomalous network entities such as nodes and edges when an entire graph structure is present. The static graphs may be plain or attributed. A graphs is said to be attributed when the nodes and/or edges of the graphs have some features associated with them. On the other hand, plain graphs consist of nodes and edges connecting the nodes. Static graph based anomaly detection is used when the structure of the graph is complete and known.

Definition 1 : Static-Graph Anomaly Detection Problem
Given the snapshot of a (plain or attributed) graph database
Find the nodes and/or edges and/or substructures that are ”few and different” or deviate significantly from the patterns observed in the graphs

Detecting anomalies for static graph based anomalies can be performed by using a state machine. A state machine is a directed graph that consists of states and state transitions. The states are represented as nodes and the state transitions are represented as edges. Using state machines for detecting anomalies has been seen in a variety of anomaly detection problems. For example, Treutniet [30] presents how a state machine was used to follow the progression of a Transmission Control Protocol

connection and detect any irregularity that occurs in the standard well defined protocol. Furthermore, Maier et al. [14] use a probabilistic deterministic timed state machine to detect anomalies in production plants. Michael et al. [15] compare the performance between using state machine and n-grams methods to detect anomalies in computer audit data. The results of their study suggest that a state machine has a capacity to represent long-term dependencies while the n-gram method performs slightly better but is slower to learn and requires a bigger amount of data when such dependencies are in question. Finally, Allen [3] researches how state machines can be used for discrete event system in manufacturing. A discrete event system is defined as a system that has defined states and defined state transitions which represent occurrence of events. Using a state machine to represent such systems leads to detecting any unusual and unallowed behavior by the manufacturing machines.

Another common way to detect anomalies in graph based data is by using graph similarity. Common ways to do graph comparison is by looking at isomorphism [31] and [17], the maximum/minimum common sub-graphs [9], [18] or the error-correction [5]. Two graphs are *isomorphic* when they have the same number of nodes and edges. Furthermore, the edge connections between the nodes of the two graphs are exactly the same. Additionally, maximum/minimum common sub graphs is calculated by checking the level of isomorphism between two graphs. The error correction method check how many edges/nodes need to be changes in order for the two graphs to be identical. The error correction method gives a solid information of how much diversity is detected between the two graphs.

Using graph similarity methods are of great advantage when the problem in question can be represented by one graph that is always followed. A disadvantages of the methods are encountered when the problem in question is represented by a graph that has multiple correct paths. For example, if the system is such that a task can be performed in multiple ways (various edge paths can be followed to perform a task), and the graph represents all the paths that can be taken. This is because the graph that represents the data that is being checked for an anomaly, represents only the one path that was followed in that scenario. In such a case, an anomaly will be detected as the graphs will never be isomorphic. As the problem of this research is a problem that represents a system in which multiple paths can be taken, using isomorphism, graph error-correction and/or maximum/minimum sub graph matching will make the anomaly detection process challenging and less accurate.

Anomaly detection on *dynamic graphs* can be also defined as temporal anomalous pattern, event or change-point detection. It is performed on time series graph data.
Definition 2: Dynamic Graph Anomaly Detection Problem
Given a sequence of (plain or attributed graphs)
Find (i)the timestamps that correspond to a change or event, as well as,
(ii)the top-k nodes, edges, or parts of the graph that contribute most to the change (attribution)

Dynamic graph based anomaly detection is not discussed in detailed as the problem this project is solving is not defined a dynamic graph. For more information about dynamic graph anomaly detection please refer to [2].

2.6.2. Challenges of graph based anomaly detection

In this section the challenges of using anomaly detection techniques will be discussed.

The anomaly detection survey by Varun Chandola et al. [6] and the graph based anomaly detection survey by Leman Akoglu et al. [2] presents few challenges that are associated with using anomaly detection techniques and graph based anomaly detection, respectively. A basic anomaly detection approach is to define a pattern/region/behavior that the data should fall into and is seen as correct. Furthermore, to declare any observation that would be seen as unusual and therefore an anomaly. This approach is very simple but there are few challenges that come with it.

Data-specific challenges

- The general challenges that come when using big data also apply on this techniques. Some of these challenges are: volume, velocity and variety of massive, streaming, and complex datasets. As collecting data is much easier now days, there are big volumes of data. Furthermore, the data

is very rich and complex and comes at a high rate.

Domain-specific challenges

- Different domains have different tolerance of anomaly. For example, a small deviation from normal in the medical domain might be a significant anomaly indicator while in the stock market domain a small division should not be seen as anomaly. Thus applying a technique developed for one domain to another is not very straightforward.

Problem-specific challenges

- Another reason for why it is difficult to specify what is seen as anomalous is that the boundary between anomalous and non-anomalous is not always very precise. This is why sometimes there can be miss classification between what is anomalous and what is not. For example, a deviation from the usual behavior might be allowed by the system that is being analyzed, and therefore is not detected as an anomaly, but some deviation from the usual behavior might indicate a fault in the system and therefore should be reported
- Another key challenge is the class imbalance. This challenge arises since anomalies are rare, only a small part of the data is expected to be abnormal
- Additional challenge in the process is explaining the anomaly detected. This means finding the root of the problem and being able to explain when and how was this anomaly caused. Presenting the results in a user-friendly form for further analysis can be a difficult task

Graph-specific challenges

- For inter-dependent objects, the relational nature of the data makes it challenging to compute the level of anomaly of the graph objects. The traditional anomaly detection assumes that the objects are independent and identically distributed while the objects in graph data have long-range correlation.
- Defining anomaly in graphs is much more diverse than in the traditional anomaly detection, given the rich representation of graphs.

The above challenges show that a general anomaly detection problem is difficult to solve. In anomaly detection there are various factors that decide which approach should be followed. Some general factors to look at are : the nature of the data, availability of labeled data, type of anomaly to be detected, etc. Different domains satisfy different combination of the mentioned factors.

2.7. Research gaps and Research question

As shown in the previous section of this chapter, Natural Language Generation techniques, nor anomaly detection techniques have not been used in the rail domain to check or explain Train Control System abnormalities. Due to the challenge of performing the log analysis by hand, this project applies Natural language generation techniques on the log files from the train control system in The Netherlands. In this research, we develop a natural language generation systems that uses two different graph representation in terms of the data retrieved from the logs of the train control system. We compare the performance based on the reports generated by the NLG system from the two different graph representations. The difference in the graph representation is explained in detailed in 4.2.1.

The goal of the project is to answer the following research question:
When reporting anomalies of a train control system, what degree of detail from the log files should the NLG system use such that the user gets a full understanding of the anomaly?

2.8. Summary

In this chapter the reader has been introduced to the Natural Language Generation techniques one could use for implementing a Natural Language Generation System. The general pipeline and all the tasks that need to be performed for a NGL system were explained in detailed. Furthermore, as the Content Determination/Signal Analysis task discussed required a anomaly detection analysis of the data,

techniques on how anomaly detection can be performed is also discusses in details. After reading this chapter, the reader is familiar with all the steps that will be taken in this research. Additionally, the ways to evaluate a Natural Language Generation system are explained. The exact techniques chosen for implementing the Natural Language Generation system in this research are explained in Chapter 4.

3

Project context

This chapter presents the reader with the project context. In Section 3.1 the background of the company is presented. Section 3.2 presents the reader with the required domain knowledge for understanding the problem. This includes an explanation of the train control system in the Netherlands as well as ASTRIS, the software that is being analyzed. Finally, the log files produced by ASTRIS are explained. In section 3.4 an overview of the problem that this thesis is aiming to solve is explained.

3.1. Background information

This thesis is done in collaboration with CGI therefore, the problem is proposed by the company.

CGI is an IT consultancy company and has a very big impact in the railway system in the Netherlands. The railway infrastructure in the Netherlands is maintained by a government task organization called ProRail. The train traffic is managed by a train control system. One layer of the system is the ASTRIS software that has been developed by CGI for their client ProRail. The software provides up to date information on the state of the railway infrastructure as well as it allows the train controller to control the traffic and the rails. More details about the train control system and ASTRIS can be found in the following sections.

3.2. The architecture of the train control system (TCS) & ASTRIS

In Figure 3.1, the structure of the train control system (TCS) is presented. The TCS in the Netherlands consist of 3 main layers: the layer which the train controller uses to send requests and receive status updates (Procesleiding), ASTRIS and BevNL which is the layer that directly controls the rails and all the rail elements and performs an additional security check.

1. Procesleiding
   The Process Leader layer (Procesleiding) is the layer in which everything gets started. A request is sent from the procesleiding to Astris Routering Component for some action to be taken (e.g. a route to be set, a switch to be turned etc.). This initialization process starts with a message (request) sent to Astris. The messages that go between the Procesleiding and Astris Routering Component are stored in one log file. That file is named ARC. In Figure 3.1 this is represented by the dark red arrow.

2. ASTRIS

   - *Astris Routering Component(ARC)* has the role to transfer this request to one of the main components. Based on what command is requested, ARC will send it to the responsible component. This communication goes through the Message oriented middleware.

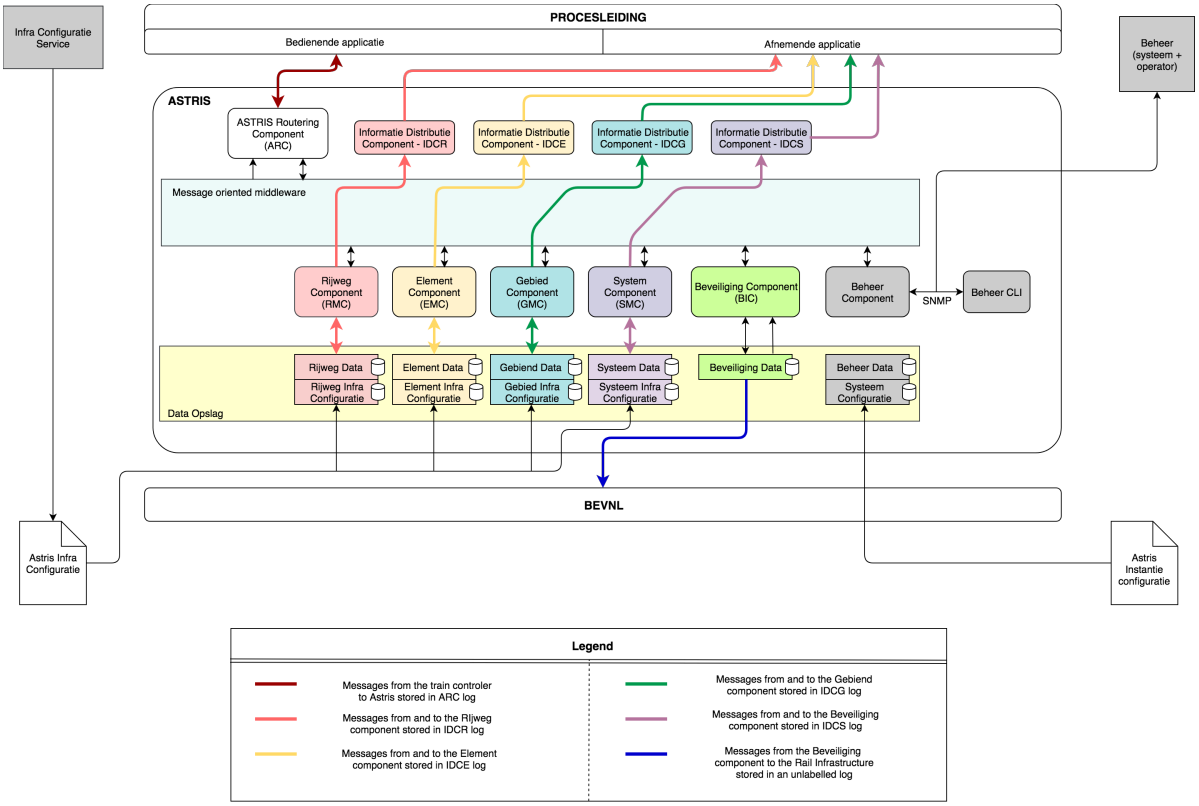

Figure 3.1: Overview of Astris

- The *Message oriented middleware - MOM* is a communication layer within Astris. Messages between the Astris components go through this layer. Not all of this messages are stored in a log file.

- The *Route Setting Component (Rijweg Component - RMC)* is responsible for setting a route. Setting a route consists of more commands that control multiple elements (e.g. turning a switch, setting the correct sign etc.) This is why this component communicates a lot with the Element Component. Each element that needs to be adjusted for a route, is adjusted such that the Rijweg Component sends a separate command to the Element component for each element that needs to be adjusted.

- The *Element Component (Element Component - EMC)* is in control of the elements such as switches, signs, sections, crosses, etc.

- The *Authority/Area Component (Gebied Component - GMC)* is responsible to give the control to the right person (e.g. this is used when there is maintenance on the rails. The gebied component then authorizes the maintenance workers to be in control of that particular area on the railway) and no trains have access to that area at that moment.

- *The System Component (Systeem Component - SMC)* is the technical component, it handles all commands that do not fall in the regular component list (the previous 3 components).

- The *Security Component (Bevieiliging Component)* communicates with the other 4 components and the BEVNL layer. They exchange information regarding a requested command that needs to be performed and Bevieiliging component checks the security measures that come with the commands. Through this component all the commands are sent to the BEVNL.

- The *Management Component (Beheer Component)* is responsible for starting the Astris software such that it starts all the other components. It continuously checks if all the other components in Astris are up and running.

3. BEVNL
   The BEVNL is the layer with the software that controls the rails and all the rail elements directly.

The 4 main components are the RMC, EMC, GMC, SMC. If everything runs smoothly those are the components that cover any type of request. This 4 components communicate with the *Information Distribution Component* through the MOM. Each important message that goes from or to this 4 components is stored in a corresponding log file - *IDCR, IDCE, IDCG, IDCS*.

This communication and the message flows that are stored are represented by different color arrows in Figure 1.

3.3. The functionality of ASTRIS

As explained above ASTRIS is one layer in the train control system. It handles the communication between the train controller and the rail infrastructure. It is a safety critical software that gets messages from the train controller for a particular command (such as, preparing the rails for a specific route a train needs to take or making sure there is no traffic on places in the rails where some construction work is being done) and checks if the command requested is safe to be performed. If so, the software sends the request to the rail infrastructure layer with requests for all the necessary actions to be performed such that the command requested by the train controller is completed.

During the communication, ASTRIS generates a high amount of data records. This data records are stored in log files. Log files are files that record either events that occur in an operating system or other software runs, or messages between different users of a communication software. The log files from Astris contain all the information about any action that has been requested/performed on the railways, all the messages that have been sent between the different components of the train control system and all the relevant status updates of the railway infrastructure. As this actions are highly technical, so are the log files. They can be used in many different ways and contribute in having a better overview of the rail traffic and potentially further improving it. They can also contribute in finding a reason behind an unwanted matter or analyzing an incident report. Furthermore, from the log files it is possible to extract key performances indicators of the train control software systems.

Currently, this log files are analyzed manually by the developers which is very inefficient and time consuming. This research aims to detect any abnormalities in the system and translate the anomalous log messages into human readable reports. Information from the log files will be extracted and reported as summaries. This will contribute to easier detection of anomalies in the software by analyzing the log files. To perform the security check on the requests from the train controller and if an action is secure, ASTRIS will pass the request to the rail infrastructure software layer (BEVNL). It also serves as a communication layer that transfers messages between the train controller and the software that controls the rails. The main role of ASTRIS is when a request sent by the train controller is received, ASTRIS check if the request satisfies all the security requirements and therefore is safe to be sent to the software that controls the rail infrastructure. Astris has all the safety rules and can ensure whether a certain action is safe to be performed or not. Therefore ASTRIS is a **safety critical** application.

3.4. Problem statement

Having the basic background knowledge of how ASTRIS works is presented such that one can understand the problem we are trying to solve.

This thesis is focusing on one of the main ASTRIS components - Route Setting Component (Rijweg Component - RMC). This component generates the IDCR log files. As explained above, this component

is concerned with requests for setting a route. The RMC component checks if a request on setting a route sent by the train controller is safe to be performed. For one route to be set, the train controller sends a request in which he specifies which route needs to be set by giving 3 main and most important information. These are the beginning sign, the end sign and a the LR (left-right) String (the LR string specifies with which directions the route needs to be set). For instance, the rails have a lot of switches that enable the trains to go in a specific direction. This directions are specified by the train controller by this LR String.

Setting the route is not a simple process and as mentioned above a lot of requirements needs to be checked and satisfied before the route is ready to be set. Firstly, the route is checked and reserved (if safe). The RMC communicates with the lowest layer of the TCS and when one stage is ready to be executed (for example all elements of the route to be reserved) the request is passed onto the lowest layer (BEVNL). There is an exact pattern that is followed during the process, and some of the steps requires back and forth communication between RMC and BEVNL. This patterns are specified in terms of visited states and attributes that are active/non-active when the system is in a particular state. They indicate how far a request has got and the status of the rails (whenever a change has occurred). To make this more understandable we created the graph that represents the normal behavior of ASTRIS. This graph is shown in Figure 4.3. A request always starts in Rest state with no attributes active, furthermore this is also the expected end state. Unfortunately, there are cases when things do not go as planned, and some unusual behavior may be noticed in the ASTRIS software. Even though this happens very rarely, it may cause in incident. When incidents happen, there is only one way to detect where the system went wrong, and that is by looking at the log files.

As every action and status is logged in this files, they are very long and reading thought them is a very difficult and time consuming task when performed by humans.

Therefore, this thesis is trying to tackle this problem by creating human readable reports about any anomalous(unusual) behavior detected in ASTRIS by analyzing the log files.

What is seen as an anomalous behavior of ASTRIS is specified by the software developers of ASTRIS. They have specified the possible anomalies in 2 categories.

- **Defined anomalies** which are behaviors of the system that are not incorrect but are rare and their occurrence may indicate that something is wrong with the system. There are two types of predefined anomalies specified by the software developers.
  - Type 1: An example of such a case is the following: The route that is trying to be set contains a switch that has a small stone stuck in the mechanism and therefore turning this switch is not possible. For ASTRIS, this is seen as a situation where the system is not able to manipulate the switch and therefore the route can not be set. So instead of taking the regular path in Figure 4.3, ASTRIS goes through states that are specified as defined anomaly (WV1 -> G3 or WV2 -> G4, this is marked by the red edge) .
  - Type 2: Anomaly of type one is defined by a state transition (IV1 -> IA1 or IV2 -> IA1). This behavior may occur when ASTRIS has performed all the security check and evaluated the request as safe. Therefore, the request was sent to BEVNL, but BEVNL evaluated this request as unsafe. This prevents the request from being executed and therefore ASTRIS goes through an unusual state transition. This may be caused be few reasons, such as a bug in ASTRIS or a 'timing' error, meaning the request was sent few seconds/microseconds earlier or later then the BEVNL was ready to accept it.

  This is why it is very important to have an autonomous system that would spot this unusual behaviors.
- **Undefined anomalies** are behaviors which are not represented in the graph. This means that the system went through a state that is not even defined in the graph. This may happen because for some reason some of the attributes do not get activated or do not get deactivated when a state transition occured. This behaviors are not seen very often, but they may indicate a error in ASTRIS.

To find the behavior of ASTRIS for one particular request, one needs to first find all the messages in the IDCR log file that are related to this request. The approach taken to detect and explain the anomalies are explained in the next Section (4).

An example message from a log file can be found in Appendix A

4

Methodology

This chapter explains the methodology for implementing a Natural Language Generation systems that uses two different representations of ASTRIS for explaining anomalies detected in log files from a train control system. Two representations of the train control system are designed and implemented in order to be able to answer the research question *When reporting anomalies of a train control system, what degree of detail from the log files should the NLG system use such that the user gets a full understanding of the anomaly*, by comparing the two reports generated by the NLG system that using the two different representations of ASTRIS.

This chapter gives a detailed explanation of the approach taken to solve the problem in question. In Section 4.1 the requirement analysis is presented. Based on interviews with the software developers we specify the requirements into functional and non-functional. In Section 4.2 the pipeline of the NLG systems is explained. All the task performed are explained step by step following the pipeline structure used in this research. In Section 4.3 a summary of the methodology is presented.

4.1. Requirement Analysis

After carefully studying the related literature, and obtaining the domain knowledge two graphs that represent the system were designed. The graphs show a representation of ASTRIS and therefore they represent the usual behavior of the software. The graphs will be presented and explained in 4.2.1.

What is seen as an unusual/anomalous behavior was defined by my CGI supervisor and other two ASTRIS software developers.

4.1.1. Interviews

The first step taken in this research was interviews with two ASTRIS developers in order to perform the requirement analysis and identify the properties of the NLG systems. The two interviewees are CGI employees that have at least one year of experience on developing and testing ASTRIS.
The first part of the interviews were designed to address the design of the anomaly detection methods. Furthermore, the second part of the interviews consisted of questions which would help specify the design of the textual explanation generated by the NLG systems. For this, the interviewees were presented with two sample reports. The difference between the two reports is explained below.

The first part of the interview consisted of presenting the participant with the graph that represented the software and the defined anomalies. This questions were asked in order to specify the functional requirements of the system. The participants were asked to look at the graph and answer the following two questions:

1. *Do you think this graph is a good representation of what you would define as normal or anomalous behavior of ASTRIS. If not, what would you change?*

2. *Could you rate the anomalies that are represented in the graph by how bad of effect they could cause in case they happen? If so, could you please label the anomalous edges with weights that would represent this bad effect factor?*

Both participants though that graph was a correct representation of what what normal/anomalous behavior of the software. For the second question, both answers given were that there is no need to give more weight to any of the predefined anomalies as all the predefined anomalies are equally important. For the interviews, two prototype reports were created. The purpose of the reports was to illustrate the expected end result of the system and get feedback on the reports from the target users.

As explained above, the second part of the interviews consisted of question concerning the design of the human readable reports. With this questions we address the non-functional requirements of the system. The difference between the two reports was that Report 1 would show all the commands performed by the system regardless if they are anomalous or not. The anomalous command could be distinguished from the non anomalous commands by a message in red bold text saying that an anomaly has been detected. Furthermore, the command that had an anomaly had contained more information in the report. Report 2 on the other hand consisted of only anomalous commands. The two prototype reports were shown to the participant at the same time and the following questions were asked.

1. *Is there some missing information. If so, please indicate what additional information you would like to see in the report.*

2. *Do you think a visualization is necessary ?*

3. *Do you think a text file with all the original log messages adds a value to this report?*

4. *Do you prefer Report 1 (showing all the commands and point out the anomaly, if any) or Report 2 (showing only the anomalous command(s))*

5. *By looking at the report, can you assess what went wrong given the information provided in the report? And if so, which factors (parts of the report) contributed ?*

Both participants answered that the exact time when the message was logged is important and therefore should be communicated through the report.

Both participants answered that a visualization and an option to download a file with all the messages related to an anomaly were not necessary.

Both of the participants also suggested that when an automatic set route command is in question, pointing that out in the report will add value.

Both participants answered that Report 2 (showing only anomalous requests) is a better way to present the log files. They though that the list of request that do not contain an anomaly is extra information and distracts them from focusing on the anomalous requests.

4.1.2. Functional requirements

Based on the problem specification and the interview outcomes, the functional requirement of the system is to generate a textual explanation if an anomaly is detected. This means that the system should recognize anomalies as defined by the software developers and output an explanation of any anomaly detected. More specifically, the goal of the system is to generate a textual explanation whenever a *defined* or an *undefined* anomaly is detected. As we are dealing with NLG systems that use two different representation of the ASTRIS system we divide this section and interpret the functional requirement of detecting anomalies for each representation. We refer to the first representation as *SM + A*, and for the second representation as *SM - A*. The difference in the representations will be explained in detail in Section 4.2.1.

Anomaly types for NLG system that uses the *SM + A* ASTRIS representation

As mentioned above, the functional requirement of this system is to explain anomalies detected. The anomalies that this system can detect are defined and interpreted below:

- Type 1 - Type 1 anomaly stands for one of the defined anomalies. The anomalous state transition is Wordt_voorbereid with attribute *InstelenRijweg* active to Gereserveerd with attribute *InstelenRijweg* active. This defined anomaly occurs when one or more elements that are part of the route did not get in the correct position. This means that the route could not be prepared, due to a failure of adjusting all switches to the requested position.

- Type 2 - Type 2 anomaly stands for another defined anomaly. The anomalous state transition is the transition from state 'Ingestelobdracht_vertuurd' with no attributes active to state 'Ingestelopdracht_afgekurt' with no attributes active. This anomaly occurs when the ASTRIS software ran all the safety checks and determined that the route request is safe to be adjusted. After this, the request was sent to the last layer of the train control system. Due to some reason the safety system in the last layer identified this route is unsafe to be set and therefore the request send from ASTRIS to the last layer was rejected. This anomaly occurs due to contradiction between the safety evaluation of the two Train control system layers.

- Type 3 - Type 3 anomaly is another defined anomaly. It is the similar type of anomaly as anomaly Type 1. The only difference is that the route request was 'Automaat' and therefore the *Automaat* attribute was active in addition to the *InstelenRijweg* attribute for both states in question.

- Type 4 - This anomaly is also a defined anomaly and it is the similar anomaly as Anomaly Type 4 but with the *Automaat* attribute active with all the states in question

- Type 5 - Type 5 anomaly is an undefined anomaly. It occurs when an undefined state is found in one of the log messages concerned with the route request. Additionally, this may occur if a state is defined but the particular state transition is not allowed. Furthermore, this may occur if the defined end state was not reached/found in the log file. The reason behind this anomaly is a bug in the ASTRIS software.

- Type 6 - This anomaly is detected when the system does not go back to the initial state. This may occur for few reasons, some of which are:

  - Lost connection and therefore a restart of the ASTRIS software occurred
  - The log file reached its size limit and therefore the messages for all the route requests are not logged.

  This anomaly type does not explicitly indicate that there was an anomaly in the system, but as explained it might be because of file size reached, but as a prevention measure, the system will still report this as an anomaly.

Anomaly types for NLG system that uses the *SM - A* ASTRIS representation

This NLG system that uses the *SM - A* representation has the functional requirement to detect and explain the anomalies in the following way:

- Type 1 - Anomaly type 1 is a defined anomaly which indicates that the system transitioned from state *Wordt_voorbereid* to state *Gereserveerd*.

- Type 2 - Type 2 anomaly is a defined anomaly that indicates that the system transitioned from state *Ingestelopdracht_versuurd* to *Ingestelopdracht_afgekurt*.

- Type 3 - This NLG system can only detect undefined anomalies when a invalid state transition occurs. Or if the system did not go back to the initial state

4.1.3. Non-functional requirements

The non-functional requirements for the two systems are very similar, as the non-functional requirements specify the information that should be communicated to the user if an anomaly is detected.

Based on the domain knowledge obtained and the interviews outcome. Each of the reports has to include the following information:

- Exact route request specification in textual form
- Exact anomaly detected explain in textual form
- Anomalous state transition specified in textual form (for NLG with *SM + A*, state and active attributes are included while for NLG with *SM - A* only states are included)
- Table in which all the anomalous messages are presented in correct order with the following details:
  - Exact time in term of seconds and microseconds when the anomalous messages were logged
  - The checksum of each anomalous message logged
  - The state and the active/non-active attributes indicated in each anomalous log message

4.2. The architecture of the NLG system

In this section the Natural Language Generation pipeline used for this project is presented. Figure 4.1 shows the pipeline visually. Furthermore, Section 4.2.1 explains the first task of the NLG systems. It explains how the logs were analyzed and which anomaly detection method was used. In section 4.2.2 the second task - Data Interpretation is presented. Furthermore, section 4.2.3 explains how the document planning task was executed. In Section 4.2.4 we explain the way the Microplnanning and Realisation task were performed in this project. The templates designed for each anomaly as well as the final step where the templates are generated in grammatically correct English language.

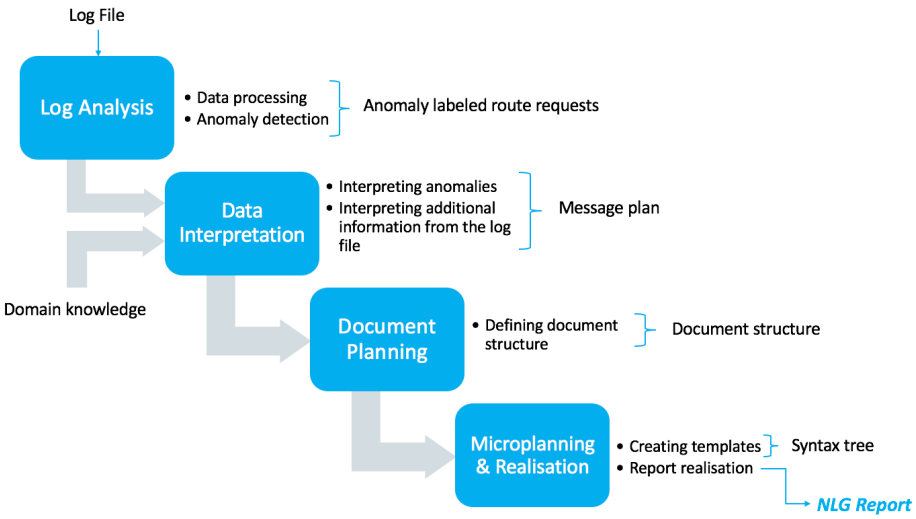

Figure 4.1: The pipeline used for the NLG systems developed in this research

4.2.1. Data Analysis

Due to the nature of the data for this research, the signal analysis task will be excluded from the pipeline and changed with a Data Analysis task. The NLG pipeline task specification suggests that the signal processing task is for numerical data only and when dealing with structured data this could be skipped.

The reason for still inducing an additional task before the data interpretation is that even thought the log files have a structured format, the data itself is not structured as the log consists of a lot of

messages which are not mapped or labeled in any way. Furthermore, the goal of this research is to detect log massages that report abnormalities in the system. Therefore the log files need to be analyzed and this abnormalities need to be detected in the first stage. Therefore with a minor modification from the original pipeline, log analysis will be performed as a first step in the module. This analysis will be performed in two steps. Firstly, the data will be processed and the log messages will be mapped. After this sub-task is performed, a state machine will be used to detect the anomalies in the TCS. Both sub-tasks are explained in detail in the next two sub-sections.

Data Processing

The goal of Data Processing is to identify the patterns of messages the correspond to the same route request.

The data used in this research is log files that are structured and presented in an xml form. As mentioned earlier this system does not give a unique identification that would show the sequence of messages concerned with each and every route request. Therefore, one of the early sub-tasks in the data processing is to add this identification to each log message. Each step of the data processing method is presented below:

1. The log files are cleaned and all the excess information are erased.

2. Each log message is parsed and stored in a csv file. During this task each message is assigned a unique ID. Each xml element within a message is parsed in a separate column. We will refer to this csv file as the *Messages csv*

3. Next, we perform data mapping based on 3 *(BeginObjectNaam, EndObjectNaam, LRString)* columns parsed from the log message. This three elements represent an identification of a route. Once the system detected which messages are from a same route, another csv file is created. We will refer to this csv file as the *Routes csv*. In each row of the csv we store the 3 elements that identify the route and each route/row is assigned a unique ID. Additionally, in a column called 'set' all the messages IDs mapped to the route are stored. This shows us which messages correspond to the exact route. Furthermore, we add a 'Messages' column that shows the number of messages detected for the route in question.

4. At this point, there are two different procedure in the data processing based on the two different representations of the train control system.

   (a) For one of the models, the next step in the prepossessing is the following: For each route stored in the Routes csv, iteration through all the IDs stored in the 'set' column is performed. For each ID in the 'set' the system refers to the *Messages csv* file and retrieves the value of the *'RijwegToestand' (Route state)* column. This column represents the state of the route at the moment the message was logged. Each route state is encoded by a upper-case symbol that is the first(or first two) letters of the name of the state in question - encoding table presented in Figure 4.2. Furthermore, the system takes the values of 6 attributes *'Automaat', 'Herroepen', 'ReserverenRijweg', 'VoorbereidRijweg', 'InstellenRijweg', 'RijwegVrijrijden'* that add extra information of the route state. The values this attributes can hold are 0 and 1, representing if an attribute was Not active or Active at the moment the message was logged. Based on the combination of active/nonactive status of this attributes, the system assigns a new symbol to the route that is concatenated to the encoding of the route state.

   Once the route state and the route attributes are encoded and concatenated, the system splits the concatenation whenever the initial state of a route is detected (the initial state is encoded as *'Ra'* which represents that the route is in Rest state with all the attributes non active.) The decision to split the route messages in this way is because each time a route request is sent by the train controller the route starts and end in this state. This step is performed as our goal is to have all the messages per route request. The train controller might request the same route to be set multiple times during the day, and we want to treat each request as a separate. This will help to have a more precise information of where the

State full name	State Encoding
Rust	R
Gereserveerd	G
Wordt_voorbereid	WV
Voorbereid	V
Instelopdracht_verstuurd	IV
Instelopdracht_afgekeurd	IA
Wordt_ingesteld	WI
Ingesteld_sein_udss	IS
Afrijden	A
IngestelOpdracht&HerroepOpdrachtVesruurd	I&H

Figure 4.2: State Encoding table

anomaly is, if any anomaly is detected.

When the system detects multiple request per same route and splits it, a new row is added and the identification of the route is copied, as well as the corresponding IDs of the messages that are split to represent one request at a time. A corresponding command ID is also assigned by incriminating the largest command ID detected in the *Routes csv* file.

Lastly, from each row the encoded concatenation is further encoded to represent each state transition. So as each message represents one state, when the messages are encoded and concatenated, the concatenation represents each state transitions the route request followed to prepare the route. The last encoding is done based on the state machine shown is Figure 4.3. This encoding is stored in the *Routes csv* in a column named *'String'* and is the value that is given to the state machine. The string represents the pattern of the state transitions per request reported in the log file.

(b) For the second anomaly detection model, this step is very similar. The only difference is that the 6 attributes used in the above explained methods are omitted. So the state transition pattern is represented only by the *'RijwegToestand'* (Route state).

Anomaly detection

To detect anomalies in the log files a finite state machine is used due to the nature of the problem. As explained above, the ASTRIS system checks all the safety critical rules, and based on that, it sets the route as requested by the train controller. The process of setting a route is performed gradually. The ASTRIS system checks for the safety requirements, and if the requirement in question is satisfied, the system transitions from one state to the other. Dependent of the type of request the system needs to follow a defined path of states in order to execute the request. The process of setting a route follows logical rules as conditions and a state transition occurs according to the a fixed set rules. State machine are used to represent a set of complex rules and condition. Each state of the state machine represents a physical condition.

As this research is trying to check if all the ASTRIS rules were followed, based on the ASTRIS requirements, a state machine is a perfect fit for the problem. All the usual and allowed state transitions of ASTRIS are implemented as rules of the state machine and the string generated from the log files is given to the state machine as input. As we want to checking if all the ASTRIS rules were followed correctly, the state machine allows us to check this in an accurate and precise way.

To summarize, the benefits of using a state machine are :

- The problem in question is about checking state transitions of the ASTRIS system, and this is exactly what a finite state machine allows
- Compared to other graph based anomaly detection algorithm (e.g. graph adjacency matrix), when one uses a state machine, and the string is rejected, retrieving the exact location and reason for rejecting the string is very straight forward. For example, if a Adjacency matrix is used

instead of a state machine, a way to check if something went wrong might be straight forward, but finding the exact reason and location for it would be more complex than when using a state machine

A disadvantage of using a finite state machine is the labor intensive work to write out all the rules/conditions for the state transitions when the state machine represents a system with a lot of requirements. Furthermore, if one wants to scale up or extend the system, this might be challenging. Even though this is a disadvantage for using a state machine, up-to-date research has shown that there are ways to let the state machine construct/learn all the rules by itself by using a machine learning algorithms and a lot of data. This will be further discussed in the Future Work Section.

Once the data is processed and we have obtained patters that represent the state transitions for every request logged, two different representations of the ASTRIS systems are created in terms of a state machine. The first representation is a state machine that uses the active/non-active attributes in addition to the state name to define the states. We call that representation a state machine with attributes. The second representation is a state machine that uses only the state names. We refer to this representation as a state machine without attributes.

State Machine with Attributes

The states in the state machine represent the state defined by the ASTRIS software. Furthermore, this state machine uses extra information (state attributes) to further describe the state. These is described in the data processing section (Section 4.2.1).

The state machine for this method is shown by the Figure 4.3. The red lines in the state machine are state transitions that represent the defines anomalies. This state transitions are shown in the graph but are not implemented in the state machine. This is because if any of those state transitions occur, the system went through a defined anomaly and this is the scenario that the NLG system should report in the final texts.

A state machine is defined by a list of its states, its initial state, and the conditions for each transition. The state machine with attributes consist of 36 states. The Rest state with all attributes nonactive is the initial and the end state. Each state transition is represented by one unique symbol. This symbols are seen as a rule for a transition to happen. The only way to reach one state from another is if this exact symbol is defined and allows that state transition to occur. The state machine takes strings as input. Each symbol in the string represents one state transition. If the sting contains a symbol that is not accepted by the state machine, the whole sting is rejected. Rejecting a string shows that the string given is violating some rules and that an unallowed state transition is present in the string.

For example when the state machine is state 'Ra' (see figure 4.3) a valid state transition would be a transition to state 'Rd' or state 'Rh'. State 'Rd' can be reached if the initial symbol of the string is **a** (see arrow label in figure 4.3), furthermore to reach state 'Rh' the initial symbol of the string has to be **b**. Then from state 'Rd' a valid state transition would a state transition to 'Gf' which would mean that the second symbol of the string is **c** and so on.

With the state machine implementation, 8 columns are added to the *Routes csv* file. Those are : 'SM_Status', 'SM_Message', 'SM_Stepwise', 'Anomaly Type', 'Anomaly Transition ID', 'First Anomaly State', 'Second Anomaly State', 'Index Of Anomaly Transition'.

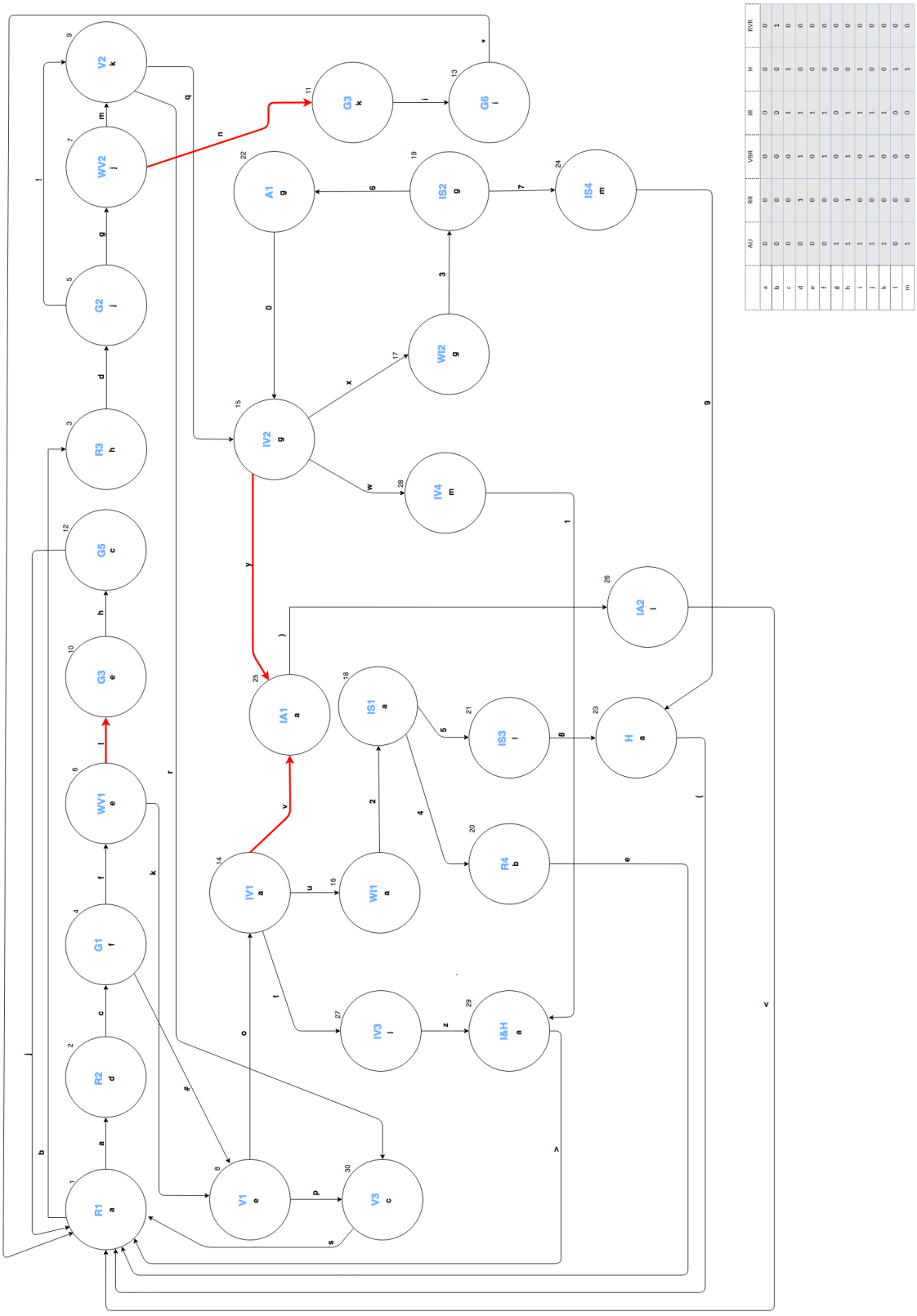

Figure 4.3: ASTRIS behavior graph

The state machine goes through the *Routes csv* file and takes the value for the the column *'String'* and runs the state machine for the string. The output of the state machine is stored in the 'SM_Status' column. If the state machine rejects the string, the stepwise function is invoked. The stepwise functions gives all the states that were visited until the anomalous transition occurred. Furthermore, an error message generated by the system points out that went wrong. After following some rules, the state machine identifies the type of the anomaly that occurred. This is stored in the appropriate tables. Lastly, when an anomaly is detected, by following the output of the stepwise function, rules of the system detect the anomalous state transition and the index of the anomalous state transition.

State Machine without Attributes

The state machine for this method is represented in Figure 4.4. In contains 11 states, which is drastically less compared to the state machine with attributes. This anomaly detection method works very similar to the state machine with attribute, except it does not consider any attributes. So the states are represented only by the state name. For this reason it is known that this anomaly detection technique is less accurate. It will miss anomalies where the state transition is correct when looking at the name of the state, but if the attributes that are active do not correspond to the state correctly, this method will not be able to recognize that. Even though we are aware that this state machine is less accurate, we use it as its has less rules and therefore it is easier and faster to implement. Furthermore, as the research question suggests, we want to evaluate if the lack of accuracy will significantly show lower performance of the NLG system. As if this is not the case, having a faster and easier method for the anomaly detection process might be preferred.

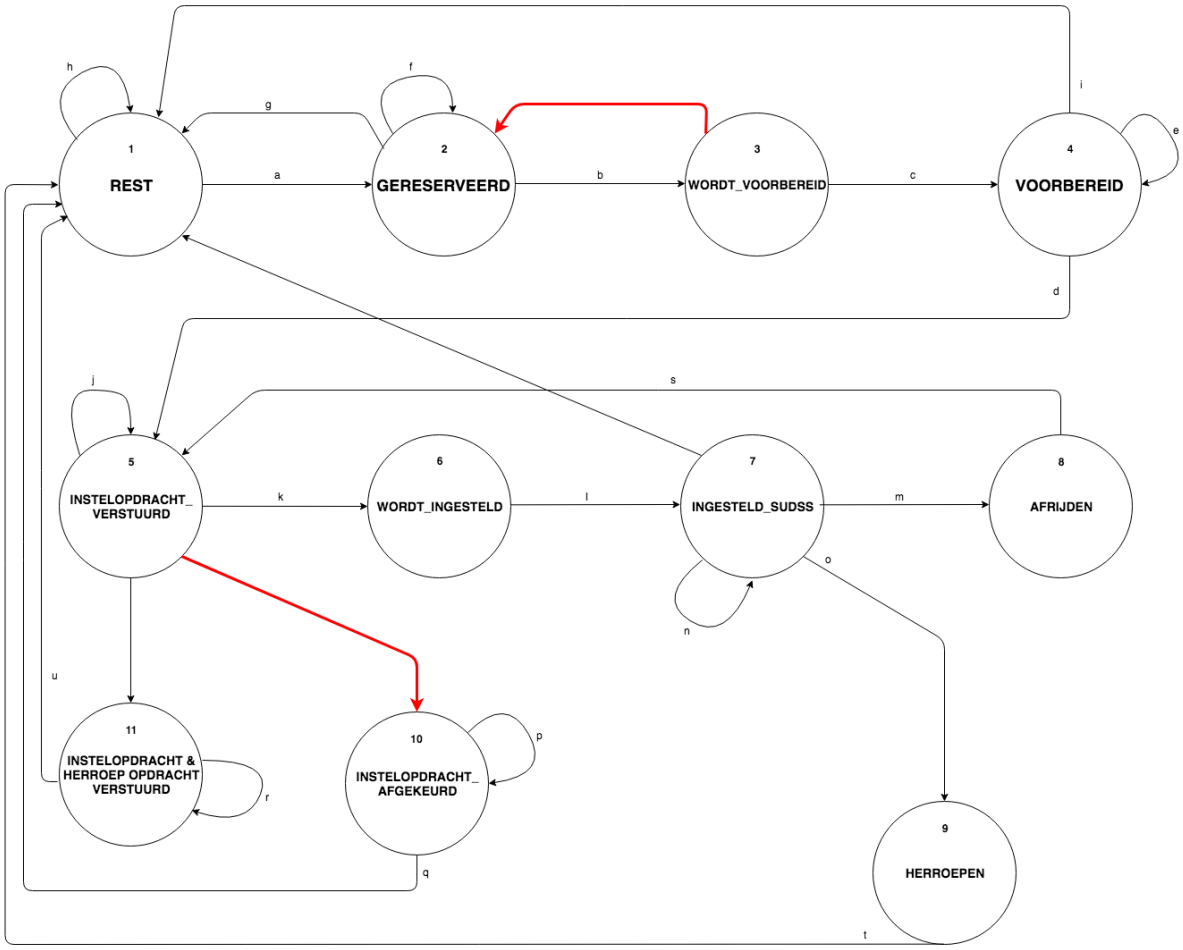

Figure 4.4: ASTRIS State Machine representation without attributes

4.2.2. Data Interpretation

The goal of the data interpretation task is to analyze and interpret the patterns and abnormalities detected in the data. This is the task of interpreting the meaning of the patterns detected and mapping the patterns and events with the additional information provided in the log files. This patterns need to be mapped into messages and relationships humans use when discussing the domain. Additionally, we interpret the exact location of the anomaly and the additional information that will be presented to the user.

For the data interpretation we follow the functional and non-functional requirements explained in 4.1.2, 4.1.3 to interpret the pattern and present the target users all required information.

Each anomaly is given same importance. This is primarily because any anomaly might cause an incident and therefore it is equally important to report it. Additionally, as explained in the requirement analysis, the software developers were asked if they would weight some of the anomalies with a higher score and the answer was that they also see all the anomalies to be equally important.

4.2.3. Document Planning

The goal of the document planning task is to decide which of the events/patterns detected will be communicated to the user. Furthermore, in this task the document structure is decided upon.

Deciding on what information to communicate

For this research deciding on which events to communicate is a straight forward task. Based on the answers from the requirement analysis and the specifications of this research topic, this project is only going to focus on communicating the anomalies. Any route request that does not contain an anomaly will not be part of the report.

Deciding on the document structure

For each log file given to the NLG system, one document will be produced. This document will contain explanation to all anomalies detected in the given log file. Additional to the summaries that explain the anomaly, a table will be displayed for each anomaly detected. The table will have some basic information about all the messages that are part of the anomalous route request. This table takes the information directly from the *Messages csv*, which is the parsed log file. The table is there to help the user see all the states the route request visited, but most importantly the exact seconds and microseconds when the message was logged. Lastly, this table also displayed the check sum value found in the log message. The time stamp and the checksum are included as this helps the used further link the anomaly to the other TCS components log files.

The document will contain a textual explanation of each anomaly detected in pair with a table which was discussed above.

4.2.4. Microplanning and Realisation

The last tasks of the NLG system is the Microplanning and Realisation. It serves as a plan on how to communicate the information the system detected in the earlier tasks. One way to perform this task is by using templates and this is the approach this research is taking.

Due to the nature of the problem, it is preferred that the text that explains a particular type of anomaly is always the same. In this way the reports generated by the system would make the explanations less confusing. For this reason, templates are used for explaning the anomalies detected. Additionally, as the reports will be used by domain experts it is important that the NLG explains the anomalies in a domain specific language. Therefore, the specific terms used by the developers are also used in the reports. As ASTRIS is developed in the Dutch language, when the text explains a particular route state, route request or attributes related to the state, the text refers to this term by the original name as seen in the log files.

Templates

The templates used by the NLG system complete two tasks in order to be lexicaly and syntactically correct. Firstly, each word is chosen carefully in order to convey full explanation of the event. As explained earlier, the words chosen need to correspond to the domain language used by the developers. The terms retrieved directly from the log files are therefore kept as they are (in Dutch) event though the texts are in English. Secondly, the explanation of the anomaly detected uses domain terminology. For example, the word 'Elements' refers to the elements of the rails such as switches, signs and so on. Furthermore, the system uses the word 'BEVNL' to refer to the last layer of the TCS. For each anomaly type a template is generated by the system. For each template a lexicalization task as well as a realization task is performed.

Lexicalization and Realization

Lexicalization is the process of adding words, set phrases, or word patterns to a language. On the other hand, realization is the process by which a text is generated based on an underlying representation. In this case the representations will be syntax trees.

Lexicalization For example when the NLG system needs to convey the specifications of a route request. Choosing the exact words for this sentence is done in the following way:
To express that there was an anomaly, we could use one of the following words: anomaly, abnormality, unusual behavior etc.

The lexicalization is performed for each sentence in order to establish the lexicon for this NLG system. Once the lexicalization is specified, we proceed with forming the phrases for each template. This is done by following syntactic rules and creating syntax trees for each template. The NLG systems uses this syntax trees to generate lexicaly and grammatically correct texts. The implementation is performed by using SimpleNLG in Java[1].

Template for anomaly Type 1 (SM + A) Below are the syntax trees used for generating a template for NLG when using *SM + A* for anomaly Type 1.
Paragraph one consist of one sentence only. This sentence specifies the exact route request where the anomaly was detected. Figure 4.5 shows the syntax tree for paragraph 1. The **X** is a placeholder for the details that are retrieved from the *Routes csv* file. Furthermore, Figure A.1 shows the syntax tree used for paragraph 2. This paragraph explains the anomaly. The **X** in this sentence is a placeholder for the unusual state transition. This states are also retrieved from *Routes csv* file. Anomaly Type 1 occurs when ASTRIS is not able to place one or more elements in the route as requested by the train controller. Lastly, paragraph 3 explains the table which shows all the messages concerned with the anomalous route request.

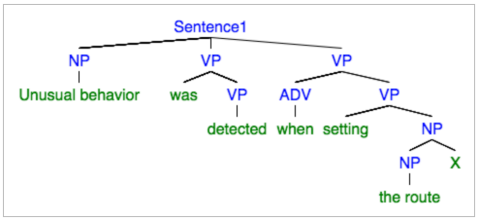

Figure 4.5: Syntax tree used for realization of paragraph 1 when explaining Type 1 anomaly with the NLG system that uses(*SM + A*). The realization of this sentence can sen seen in Figure 4.8.

Template for anomaly type 2 (SM + A)
The template for explaining anomaly type 2 when using SM+A, starts with the same sentence as anomaly type 1. Paragraph 2 consist of 3 sentences. As identified earlier, anomaly type 2 is a defined anomaly which indicates that ASTRIS identified a route request as safe to be performed, but the last layer in the TCS rejected the request. In the template this is explained with domain specific terms. The third paragraph is the same as the template for anomaly Type 1. This is because the last paragraph explains the information provided in the table that is presented after each textual explanation.

Template for anomaly type 3 (SM + A)
This template is almost exactly the same as the template for anomaly Type 1. The only difference is in the first sentence which give the specification about the route request. The difference is that when this anomaly type occurs, it is know that the route request was automatic. In the requirement analysis the ASTRIS developers explained that this is an important information and they would like have it pointed out in the textual explanation

Template for anomaly type 4 (SM + A)
This template is almost the same as the template for anomaly Type 2. The only difference is in the first sentence of the template. This template indicates that the route request sent by ASTRIS was automatic. Again, this has been decided upon based on the request of the developers of ASTRIS.

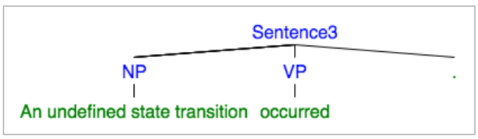

Figure 4.6: Syntax tree used for realization of sentence 3 for explaining anomaly type 5(undefined anomaly) for both NLG systems

Template for anomaly type 5 (SM + A)
This is the last type of anomaly for the system that uses a state machine with attributes in the log analysis task. Anomaly Type 5 is the undefined anomaly. Therefore the exact cause of this anomaly can not be specifically explained. Even though that is the case, the text still explains the two possible reasons on why this route request were detected as anomalous. This template, as the previous 4, has 3 paragraphs and a table for each anomaly Type 5 detected.

The syntax tree (from the second paragraph that explains the anomaly) for this template can be seen in Figures 4.6, 4.7 and A.2

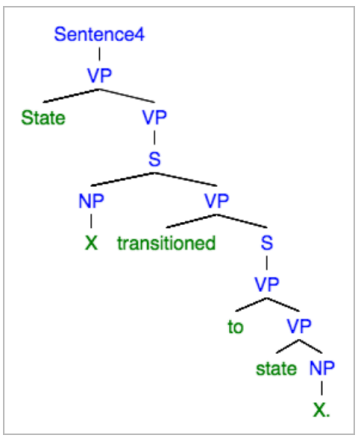

Figure 4.7: Syntax tree used for realization of sentence 4 for explaining anomaly type 5(undefined anomaly) for both NLG systems

Template for NLG that uses SM - A
The templates used for the NLG system that uses the state machine without attributes as log analysis method are very similar. As the state machine does not use state attributes to detect the anomalies, the attributes are excluded from the textual explanation of the anomalies. Everything else stays the same.

Additionally, as this system does not consider the attributes, as explained above, there are only 3 types of anomalies that the system can detect. Those are the Type 1 - defined anomaly when

state Wordt_voorbereid transitions to Gereserveerd, Type 2 - defined anomaly when state Ingestelopdract_vertuur transitions to Ingestedopdracht_afgekurt. And finally, Type 3 which is an undefined anomaly.

Figure 2.4 shows an example explanation of anomaly Type 1 generated with both NLG systems. The difference is marked with red. The words marked with red are in the report of the NLG system that uses the *SM + A*, but not in the report of the NLG system that uses the *SM - A*

SM + A

Unusual behavior was detected when setting the route 62:76:LLRR.

One or more elements did not reach the requested position for the route. Consequently, state 'Wordt_voorbereid [IR]' transitioned back to 'Gereserveerd [IR]' ,instead of transitioning to the expected state 'Voorbereid [IR]'.

The table below shows all the states visited and the attributes that were active in the particular state (1 - Actief, 0 - NonActief). Additionally, the time when a state was visited is indicated in the third column. The unusual state transition is the transition from row 4 to row 5.

Row	Checksum	Seconden	Microseconden	RijwegToestand	AU	RR	VR	IR	H	RVR
1	701095711	1554738769	979152	Rust	0	0	0	0	0	0
2	2780274026	1554738772	690953	Rust	0	1	1	1	0	0
3	923368117	1554738772	719985	Gereserveerd	0	0	1	1	0	0
4	3584276470	1554738772	742675	Wordt_voorbereid	0	0	0	1	0	0
5	2289598487	1554738774	204332	Gereserveerd	0	0	0	1	0	0
6	2157779888	1554738776	353584	Gereserveerd	0	0	0	1	1	0
7	2163222222	1554738776	391240	Rust	0	0	0	0	0	0

SM - A

Unusual behavior was detected when setting the route 62:76:LLRR.

One or more elements did not reach the requested position for the route. Consequently, state 'Wordt_voorbereid' transitioned back to 'Gereserveerd', instead of transitioning to the expected state 'Voorbereid'.

The table below shows all the states visited and the attributes that were active in the particular state (1 - Actief, 0 - NonActief). Additionally, the time when a state was visited is indicated in the third column. The unusual state transition is the transition from row 4 to row 5.

Row	Checksum	Seconden	Microseconden	RijwegToestand	AU	RR	VR	IR	H	RVR
1	701095711	1554738769	979152	Rust	0	0	0	0	0	0
2	2780274026	1554738772	690953	Rust	0	1	1	1	0	0
3	923368117	1554738772	719985	Gereserveerd	0	0	1	1	0	0
4	3584276470	1554738772	742675	Wordt_voorbereid	0	0	0	1	0	0
5	2289598487	1554738774	204332	Gereserveerd	0	0	0	1	0	0
6	2157779888	1554738776	353584	Gereserveerd	0	0	0	1	1	0
7	2163222222	1554738776	391240	Rust	0	0	0	0	0	0

Figure 4.8: Example of type 1 anomaly explained with both of the NLG systems

4.3. Summary

In this section all the steps taken for implementing an NLG system using two different representation is explained in detailed. Below a overview of the methods is summarized.

The first step taken in this research was obtaining the domain knowledge. Next, a requirement analysis was performed in a form of interviews. From the requirement analysis, we could observe what the users expected from the NLG system. A representation of ASTRIS was constructed in two ways. One was a graph that represents the system states with more information (state name + active attributes), whereas the other graph represented the system only in terms of state names.

The next step taken was the data analysis. Translating and encoding the log file into strings that represents the behavior of the route requests. Once the data was processed and each request found in the log files was in a string representation, we implemented the two graphs as state machines. The state machines outputs a label for each request. The labels would represent the behavior of the request. If there was an anomaly detected, the label would represent the type of the anomaly, while if the request was not anomalous the system would label the request as such.

Furthermore, all the anomalies were interpreted and mapped to additional information from the log files. Next, the structure of the document was decided upon.

Lastly, templates were created based on the information the system will include in the human generated text. The templates were represented by syntax trees to insure syntactically and grammatically correct sentences. The syntax trees were realized by using a realizer. Finally, the output of the realization step was the human readable reports.

5

Evaluation

Once the NLG systems were implemented, we had an NLG system that generates summaries based on log analysis that uses state machine with attributes system representation and a NLG system that generates summaries based on log analysis that uses a state machine without attributes system representation. In this chapter we evaluate the two different reports. The reports are compared in terms of user performance and user preference. To do this we perform a case study in which a sample of target users are presented with the two NLG reports. The users are asked to answer few question which will help us answer the two hypothesis of this research.

5.1. Evaluation methods

When NLG systems are developed, the main measure that one needs to evaluate are the readability, accuracy, and usefulness. As discussed in the related work section (2.5), there are three main techniques for evaluating NLG systems. Those are:

- Task Performance
- Human Rating
- Metric

In this research we use task performance and human ratings to evaluate the two NLG systems created.

Task Performance Task performance metric is used as it is seen as the most rigorous evaluation metric for NLG systems. It has been discussed that task performance is the best way to evaluate NLG. The advantages of the method are that the target users get to see the reports and answer questions that they would need to answer in real life when the NLG system is put in use. Therefore, the understanding the user acquired based on the reports presented is evaluated in a very pragmatic way. For this research, the participants are presented with the generated reports and asked questions about the text they read. This method directly measures how well the desired message is communicated to the users based on the answers they gave to the questions after they read the reports.

Human Rating The human rating method is seen as the second best way to evaluate an NLG system. This is because the target users are directly asked to express their opinion on the reports presented. Human rating evaluation is performed by asking the users to rate the usefulness, readability and accuracy of the report on a likert scale. Another way to perform human rating evaluation when one needs to compare two reports, is to ask the users to choose a preferred report base on the 3 above mentioned measures. As this research compares two different NLG system, the participants are asked to choose a preferred report. The exact questions are show in Section 5.3

It is important to mention that the Metric evaluation method is not possible for this research, as a gold-standard report needs to be available for this type of evaluation. A gold-standard is a text that has been generated by an already existing NLG system or a text that has been written by a human, based on the data used by the NLG system that is being evaluated. As there is no existing NLG system that produces anomaly detection summaries based on train control system log files, nor there is any human written reports based on the log files, a gold-standard report is not available for this research.

The task performance and the human rating methods have been used to answer the hypothesis which will be introduced in the Section 5.4.

5.2. Independent variables

The two NLG systems are evaluated based on the reports they produce. In the evaluation, we compare the two different reports that are generated based on the same log data. The independent variables in this evaluation are Report types. We perform a within-subject experiment by presenting the users with two reports at the same time.

- Report 1 is the report generated by the Natural Language generation system that uses the State Machine with attributes
- Report 2 is the report generated by the Natural Language generation system that uses the State Machine without attributes

Furthermore, the reports consists of multiple cases (anomaly types) which are specified as additional independent variables in this evaluation. Based on the representation that the NLG system uses (State machine with attributes or state machine without attributes), a different number of cases is presented in each report. Each case that is part of the report is chosen as they explain different anomaly types. The decision of using the specific cases was made as we wanted to have each defined and undefined anomaly type that can be detected by the more detailed representation (State machine with attributes) at least once. As the NLG system that uses a state machine without attributes can detect less anomaly types compared to the NLG system that uses the state machine with attributes, therefore Report 2 had one case less.

All the cases presented in the reports and the anomaly detected by each system is explained. In bold letters, the label given but the each of the state machine is also shown.

- Case 1 :
  - *SM + A* - defined anomaly - anomalous transition : *Wordt_voorbereid* (InstellenRijweg attribute active) -> *Gereserveerd* (InstellenRijweg attribute active) **Type 1**
  - *SM - A* - defined anomaly - anomalous transition : Wordt_voorbereid -> Gereserveerd **Type 1**
- Case 2 :
  - *SM + A* - defined anomaly - anomalous transition : Ingestelopdracht_versuurd (no attributes active) -> Ingestelopdracht_afgekurt (no attributes active) **Type 2**
  - *SM - A* - defined anomaly - anomalous transition : Ingestelopdracht_versuurd -> Ingestelopdracht_afgekurt **Type 2**
- Case 3 :
  - *SM + A* - defined and undefined anomaly - anomalous state transition : Ingestelopdracht_versuurd (Automaat attribute active) -> Ingestelopdracht_afgekurt (Automaat attribute active) is a defined anomaly by itself, but more importantly state Ingestelopdracht_afgekurt with Automaat attribute active is not defined in the system and this is an undefined anomaly **Type 5**
  - *SM - A* - defined anomaly - anomalous transition : Ingestelopdracht_versuurd (Automaat attribute active)-> Ingestelopdracht_afgekurt (Automaat attribute active) **Type 2**

- Case 4 :
  - *SM + A* - undefined anomaly - system did not return to initial state **Type 6**
  - *SM - A* - undefined anomaly - system did not return to initial state **Type 3**
- Case 5 :
  - *SM + A* - undefined anomaly - anomalous state transition - Rest (no attributes active) -> Rest(no attributes active) is an undefined anomaly as the system should never log messages if no change in the state or the attributes have been made **Type 6**
  - *SM - A* - anomaly not detected by this NLG system - this system does not recognize this as an anomaly as it does not consider the attributes in which case transitioning from Rest -> Rest is allowed **No anomaly**

An full example of both Report 1 and Report 2 can seen in A.3

5.3. Dependent variables

In this project we are measuring the performance and the preference of the two reports.

To avoid order biases, the Latin Square experiment design was used. The Latin Square technique suggest that when there are multiple representations or scenarios, each report should present all the possible scenarios in a different order. Following the Latin Square rule, each participant saw a report that was unique. The uniqueness was the order in which the cases explained above were ordered. This was done for every participant, but it is important to point out, that the order of the cases was kept the same between the two reports (NLG with *SM + A* and *SM - A*) presented per participant. For example, if the case order for the first participant was case 1 , case 2, case 3, etc. , this case order was followed for both Report 1 and 2.

Furthermore, for half of the participants the report generated by the NLG system that uses *SM + A* was placed on their hand-right side and the report generated by the other NLG system was placed on their left-hand side. For the other half of the participants, the reports were placed the other way around. For clarity, we refer to the report generated by the NLG system that uses *SM +A* as Report 1 and the report generated by the NLG system that uses the *SM - A* as Report 2.

We measure the **performance** of the reports by using the task performance technique. Two questions were asked after each anomaly explanation. The questions were designed to measure the general understanding of the texts the participants were presented.

- The first question is : *Which unusual behavior occurred?*. This is an open answer question where the participant needed to explain what went wrong based on the explanation read and the table provided.
- The second question is : *Please mark in the table the messages that correspond to the explained unusual behavior?*. This question measures if the participant will be able to precisely spot the anomalous transition based on the explanation provided.

These two questions were asked in a written form after each case in the report.

Furthermore, to measure the **preference** between the two reports we use the human rating evaluation technique. The participants are asked to answer 3 multiple choice questions as well as one open answer question where the participants are asked to elaborate on their answer choice of the multiple choice questions. The questions used for the preference measure are the following:

- Please indicate which report you thought was easier to read?
- Please indicate which report was more accurate?
- Please indicate which report was more useful for you?

These 3 questions are measuring the readability, accuracy and usefulness of the reports.

5.4. Hypothesis and Measures

For this research two hypothesis have been tested in order to answer the research question. The hypothesis compares the two NLG systems in terms of user performance and preference.

Hypothesis I : *Report 1 is more understandable than Report 2*

Hypothesis I measures the understandability the participants gained after they read the reports.

Hypothesis II : *Report 1 is preferred over Report 2*

Hypothesis II measures the preference of the participants over the two reports. The Preference is measures in terms of readability, accuracy and usefulness.

5.5. Materials

The materials used for the case study were the reports generated. Furthermore, a consent form was prepared for each of the participants in order to ensure the ethical clearance of the study.

Reports

As explained in the Independent variables section (5.2), two types of reports were generated for each participants. This was done following the procedure explained in Section 5.3. The questions asked during the case study were added to the reports after each case was explained. An example of the Report 1 is presented in Appendix A.3, followed by an example Report 2 in Appendix A.4.

Consent form

As suggested by most researchers, before the case study took place, a consent form that explains the case study, its purpose and the way data will be stored and used was created. This consent form was approved by the TU Delft Ethics committee before the participants were invited to take part in the study.

5.6. Procedure

During the case study each participant was presented with Report 1 and Report 2. To tackle the challenge that participants may perform better if they read one of the reports first, the participants were presented with both reports at the same time.

For every participant the case study was performed in the same way. The steps taken were the following:

- Each interview started with a brief explanation of the general goal of using an NLG system to analyze the log files.
- Then, each participant received the consent form
- If the participant agreed and signed the consent form, the following two ”warm up” questions were asked : *What is your role in ASTRIS?* & *For how long have you been working on ASTRIS?*
- Next, the participants were asked to imagine a scenario where few incidents occurred and one of their teammates asked them to look at the NLG generated reports and explain what went wrong. The two reports were placed in front of the participant. The participants were given as much time as needed to answer each question asked in the report. All the questions and answers were given in a written form.
- Lastly, the participant was given another paper with the three multiple choice questions and the open answer question in which they reason their answer of the multiple choice questions.

5.7. Results

The data used to evaluate this research is the data obtained from the case study. We briefly discuss the sample of participants that took part in the case study. This can be found in 5.7.1. Next, we discuss the two hypothesis in terms of the results obtained from the case study. We first present the results from the overall performance (H1) of the participants after which we discuss the results shown. Next, we present the performance results for each individual case and discuss them. The same order is followed for discussing H2. The preference measure results are presented and discussed.

5.7.1. Participants

The main challenge of the evaluation for this research is forming a sample of target users. This is due to the very domain specific problem in question. To accurately evaluate this research, all the participants need to have appropriate domain knowledge. This means the participants should be familiar with the ASTRIS software and the meaning of the log files that are generated by the ASTRIS system.

A list of 20 people from CGI or ProRail that are/were involved in the ASTRIS project was suggested by my company supervisor. The limited number of participants was as the CGI ASTRIS developers and testers team consist of 8 people only. From the 20 people suggested, 16 people responded and took part in the case study.

5.7.2. H1 : Report 1 is more understandable than Report 2

We are dealing with two dependent ordinal data sets. The data sets contain the answers of the question *Which unusual behavior occurred?* for Report 1 and Report 2. This question was asked after each case explanation. The answers were given in a textual form. To perform the statistical analysis we use a coding schema to classify and give a score to each answer. The coding schema used is shown in Figure 5.1. The schema help to classify each answer in one of the categories based on the correctness of the answer. In Table 5.1 we present the descriptive statistics of the overall performance of the reports, while in Table 5.3 we present the descriptive statistics for each case separately. Furthermore, in Table 5.2 we present the Wilcoxon signed rank test results from the overall performance data, while in Table 5.4 we present the Wilcoxon signed rank test results per case. After this, we discuss the overall results and by case results in the Discussion subsection.

Coding Schema

Each category is given a value based on the correctness of the answer. The categories were assigned a value in the following way:

- Answers from category *Excellent'* were assigned **1** point
- Answers from category *'Good'* were assigned **0.75** point
- Answers from category *'Average'* were assigned **0.5** point
- Answers from category *'Satisfactory'* were assigned **0.25** point
- Answers from category *'Poor'* were assigned **0** point

Furthermore, if multiple answers from a participant were missing or could not be assigned to any category, the participant was eliminated from the sample size.

By using this method, two participants were eliminated.
The first eliminated participant had given unclear answers, such as : *"duplicate of ?"* to case 3. Furthermore, the participant did not answer 4 of the questions.

The second eliminated participant had given answers such as :*"The same as described, well-formulated"* for the majority of the questions. From the given answers we can interpret that the participant thought the text was explaining the anomaly well, but we can not put the answers to any category as the level of understanding the participant gained after reading the reports it is not clear.

	Excellent	Good	Average	Satisfactory	Poor
Case 1	An element did not reach requested position	Route not set OR State reverts to Gereserveerd	WV -> G	Transition to wrong state	No answer OR Anything that does not fall in the other categories
Case 2	BevNL/the security system denied the route request	Route not set OR ASTIS request not accepted	IS_V -> IS_A	Transition to wrong state	No answer OR Anything that does not fall in the other categories
Case 3	BevNL/the security system denied the route request and the Automaat attribute was not deactivated	ASTRIS request not accepted	BevNL/the security system denied the route request and the Automaat attribute was not deactivated, but mention that the attribute fault was noticed because of Report 1 OR IS_V -> IS_A	Transition to wrong state	No answer OR Anything that does not fall in the other categories
Case 4	Missing messages(size limit reached or system restart)/ system does not go back to Rest	Missing messages	System stays in WV	No end state OR No state Rust at the end OR No unusual behavior	No answer OR Anything that does not fall in the other categories

Figure 5.1: Coding schema used to categorize and quantify answers

Once all the answers are coded we perform an analysis that compares the performance of the participants per case. Case 5 which is explained in 4.1.2, was eliminated from the dataset. It was eliminated as the case occurred only in Report 1. More importantly, all the participants had given a correct answer to the question and therefore scored one point. For this reason, the answers to this case do not add any useful information when comparing the two reports and are therefore are not used.

Statistical Results Overall Reports Performance

Table 5.1: Report 1 & 2 Descriptive Statistics Overall

Report Type	n	mean	median	sd
Report 1	13	.75	1	.3465516
Report 2	13	.6826923	.75	.3288251

Table 5.2: Wilcoxon Sign Rank Test Report 1 vs Report 2

Test statistics (Z)	Asymptotic significance (p)
1.552	0.120

Discussion overall Reports performance

The overall performance for the two reports is presented in Figure 5.1. The Table shows the mean, median and standard deviation for Report 1 and Report 2 based on all the case. Furthermore, Table 5.2 shows the report performance comparison based on the *Wilcoxon signed rank test*.

The *Wilcoxon Sign Rank Test* indicates that Report 1 and Report 2 ranks are statistically not significantly different, with Z=1.552 and p=0.120

Statistical Results per Case Reports Performance

Table 5.3: Report 1 & 2 Descriptive Statistics by Case

		Report 1			Report 2		
Case #	n	mean	median	sd	mean	median	sd
Case 1	13	.8653846	1	.2816504	.8076928	1	.3252218
Case 2	13	.7884615	1	.3202563	.8461538	1	.2802243
Case 3	13	.7692308	1	.3602883	.4807692	.5	.1898886
Case 4	13	.5769231	.5	.3870914	.5961538	.5	.3755338

Table 5.4: Wilcoxon Sign Rank per Case

Case #	Test statistics (Z)	Asymptotic significance (p)
Case 1	1	.3173
Case 2	-1.413	.1576
Case 3	2.598	.0094
Case 4	-1	.3173

Discussion per Case Reports performance

Case 1

The *Wilcoxon Sign Rank Test* indicates that Report 1 and Report 2 ranks are not statistically significantly different for case 1 with **Z = 1 and p = 0.3173**. This means that when the participant was presented with an explanation of the case in Report 1 and Report 2, the level of performance was similar as it can be observed base on the mean performance for each report. In fact, the mean performance for Case 1 Report 1 is **86%** and **79%** for Report 2. This shows that Report 1 performed slightly better. Furthermore, based on the mean performance we can conclude that the participants got a good understanding of this case regardless of the report shown. Similarly, in terms of the median(**1**) we can observe that both Report 1 and 2 give a solid performance.

Case 2

Similarly as for Case 1, the participants showed very similar level of understanding for Case 2 when presented with Report 1 and Report 2. The *Wilcoxson Sign Rank test* indicates that Report 1 and Report 2 ranks are not statistically significantly different for case 2 with **Z = -1.413 and p = 0.1576**. Furthermore, based on the mean performance for this case, we can conclude that the participants gained a good understanding. Report 1 performed slightly worse than Report 2. Report 1 had obtained **79%** mean performance while Report 2 obtained **85%**.

Based on the statistical analysis the difference in performance between report 1 and 2 for this cases is not significant but the performance is high for both cases. We can observe two reasons behind this results.

- Firstly, it is important to mention that the difference in the explanation of this two cases is very small. The only difference in the explanation is that Report 1 presents the attribute that were active next to the state that was part of the anomalous transition. Due to the close similarity of the explanation presented by the two reports, the performance is not significantly different.
- Secondly, this two cases explain a defined anomaly. This means that the anomalous transition that occurred is allowed by ASTRIS but the state transition indicates that something had gone wrong. As the transition is allowed by ASTRIS we can explain the anomaly more precisely and the participants score a high performance for this two cases.

Case 3

Contrary to the Case 1 and 2, the *Wilcoxon Signed Rank Test* indicates that Report 1 ranks are significantly higher compared to Report 2 for Case 3. The Z value of the Wilcoxon Sign Rank test is **2.598** with a significance of **p = 0.0094**. This can be also observed from the mean and the median for this case. Report 1 has a mean performance of **77%** with a median of **1**, while report 2 has a mean of **48%** and median of **0.5**. Based on the statistical results we can conclude that for case 3, participants get a better understanding of the anomaly when looking at Report 1.

From the statistical analysis we can see that the performance of Case 3 for the two reports is significantly different. Case 3 explains two anomalies, more precisely this case has both a defined and an undefined anomaly. Due to the train control system representation used, detecting the anomalies for the NLG system that generates Report 1, the anomaly detected is undefined. While the NLG system that generates Report 2 can only detect the defined anomaly in this route request. For this reason the explanation of this case is different in Report 1 and 2. Having a low performance in Report 2 for this case, indicates that the participants do need a more detailed explanation and a pointer to the attributes in order to fully understand what went wrong.

Additionally, as motioned in the 5.3 the order in which the participants were presented with the reports varies. So, 7 out of 13 participants looked at Report 1 first. 4 out of the 7 participants had explicitly mentioned that they get full understanding of the case because the saw Report 1 first. This is why in the coding schema we classify the answer as an 'Average' answer if the participants answer the question in Report 2 correctly but mentioned that that was due to the knowledge obtained by seeing Report 1 first.

Case 4
Lastly, for case 4 we observe that the difference in the performance between Report 1 and 2 is also not significant. The *Wilcoxon sign Rank Test* for this case indicates that there is no statistically significant difference in performance between Report 1 and Report 2. The test statistics from the Wilcoxon Sign Rank test is **Z = -1** with a statistical significance of **p = 0.3173** This can be also observed by looking at the mean performance of Report 1 which is **58%**, while the mean performance for Report 2 is **60%**. Furthermore, an interesting observation for this case is that the overall performance for this case is lower that the other cases. It is important to mention that for both reports this case was explained exactly the same. This is because case 4 explains a route request where the ASTIS system did not return to the initial state. Both NLG systems detect and classify this anomaly as an undefined anomaly. This type of anomaly we have discussed in 4.2.1 that this behavior is not necessarily anomalous because this may indicate that the ASTRIS system was restarted or the log file reached a size limit. This is explained in the NLG reports. The lower performance for this case in both Reports may be because some of the participants did not agree that this is anomaly. A solution to this is proposed in the 6.4.

5.7.3. H2 : Report 1 is preferred over Report 2

As explained earlier, the preference measure for Report 1 and Report 2 were obtained by asking every participant to indicate there preferred Report in term of readability, accuracy and usefulness through multiple choice questions. In addition to the three multiple choice questions, the participant was asked an open and question to explain his/her choice on the previous three questions. The open end question was used to obtain a deeper understanding of the preference choices made by the participants.

The first multiple question measures the *Readability*, the second question measures the *Accuracy* and lastly the third question measures the *Usefulness* preference between Report 1 and 2.

The answer options given to the participants was a choice between Report 1 and Report 2. Even though this was the case, some participants added an answer option of indifference between the two reports. Even though this was originally not planned, we take the answers as given by the participant and add this answer choice in out analysis. The data set consist of 14 participants as explained in the previous subsection.

To analyze the data obtained from the human rating questions we count the preference of each Report in terms of the three measures. Furthermore, we count the overall preference per report.

Statistical Results

The table below show the results from the multiple choice questions. A percentage analysis is applied to create a contingency table for report preference.

Table 5.5: Report Preference Measures Results

Measure	Report 1	Report 2	Report 1&2
Readability	54%	23%	23%
Accuracy	84%	8%	8%
Usefulness	77%	15%	8%
Overall	72%	15%	13%

Discussion

Readability

The preference in term of **readability** is presented in row 1 in Table 5.5. It can be observed that **54%** of the participants prefer Report 1, **23%** of the participants prefer Report 2 and **23%** of the participants are indifferent between the two reports in terms of readability.

Accuracy

The human rating preference results for **accuracy** are shown in row 2 in Table 5.5. **84%** of the participants answered that Report 1 is more accurate than Report 2. Furthermore, **8%** of the participants answered that Report 1 is more accurate and finally the rest of the participants (**8%**) did not notice any difference between the two reports in terms of accuracy.

Usefulness

In term of **usefulness**, **77%** of the participants answered that Report 1 was more useful than Report 2. Furthermore, **15%** of the participants indicated that Report 2 was more useful then Report 1 and lastly, **8%** of the participants did not see any difference between Report 1 and Report 2 in terms of usefulness. This can be observed in row 3 in Table 5.5.

Overall Preference

Finally, we present the **overall preference** of the Reports in the last row of Table 5.5. It can be observed that **72%** of the participants prefer Report 1, **15%** of the participants prefer Report 2 and finally **13%** of the participants did not indicate a preference between the two reports.

In the next paragraph we analyze the preference measure results based on the answers given by the participants for the open end questions. These type of questions are often refereed to as comments questions in literature. As indicated earlier, we can observe that generally, the preference in terms of readability is higher for Report 1, but compared to accuracy and usefulness, the readability measure of Report 1 is the lowest. We discuss this performance result based on the comments given by the participants. As shown and indicated in Figure 4.8 the difference between Report 1 and 2 in terms of the text presented is that Report 1 shows the active attributes next to where the anomalous state is shown. On the other hand, this active attributes are not shown in the explanations of Report 2. Few of the comments from the participants stated that, they are not used to seeing the active attributes next to the state in question. Some participants indicated that this is useful for them, but as they are not used to it, the readability when the anomaly is explained in this way is worse. Furthermore, some participants such as project managers or product owners that are not used to seeing the log files on daily bases, indicated that the attributes do not add any value for them, and it even confuses them. Based on this answer, it can be concluded that having the attributes as part of the textual explanation is useful when people are familiar with the representations. Additionally, people with deeper understanding of ASTRIS that are used to seeing the attributes thought that the readability is not negatively affected when the attributes are included in the text.

On the other hand an interesting observation is that few participants explained that when the attributes are presented as in Report 1, it is easier to relate the textual explanation to the table provided after each explanation and that indeed the attributes help them understand the anomalies especially

when the anomaly detected is undefined.

5.8. Summary

In this section the two hypothesis will be answered base on the results presented in section 5.7.

5.8.1. Hypothesis 1

As stated in Section 5.4, the first hypothesis of this research is *H1:Report 1 is more understandable than Report 2*. We answer this hypothesis based on the *Task Performance* results.

Based on the *Wilcoxon signed rank test* the overall performance of the two reports was not significantly different, and therefore this hypothesis is rejected.

When performing the performance analysis per case, we can observe that there is no significant difference for Case 1,2 and 4, but for case 3 a significant difference was noticed. Based on the qualitative analysis we have observed that Report 1 does perform better. As the sample size of this research is small, and due to the results of the per case analysis, we believe that this hypothesis is worth studying further.

5.8.2. Hypothesis 2

The second hypothesis of this research is *H2: Report 1 is preferred over Report 2*

We answer this hypothesis based on the *Human Rating* results. H2 is accepted based on both the qualitative and quantitative analysis. Furthermore, the quantitative results shows that Report 1 is preferred base on *Readability, Accuracy and Usefulness* as well as for the overall analysis. Therefore, we can conclude that Report 1 is preferred over Report2.

5.9. Answering Research Question

The research question that this research answers is *When reporting anomalies of a train control system, what degree of detail from the log files should the NLG system use such that the user gets a full understanding of the anomaly?*

Based on two hypothesis answered in Section 5.7 we answer the research question. When developing an NLG system for a train control system, one should use a more detailed representation of the system and extract more information available in the log file, specifically, the state names and state attributes. We can conclude that the NLG system that generates Report 1, which uses all the main information provided in the log file, the target users get a better understanding of the anomaly explained. In the case of this research we can establish that the users perform better and prefer to read the report by the NLG system that uses the attributes over the NLG system that does not use the attributes.

6

Discussion and Future Work

In this chapter we summarize the research work of this thesis. Furthermore, we present the limitations and conclude the findings. At last, we give our suggestions for improvements and future work.

6.1. Project Summary

6.1.1. Research goal

Natural Language Generation techniques have not yet been used in the rail domain. The goal of this research is to design, implement and evaluate a Natural Language Generation system that can produce human readable summaries for explaining anomalies detected in the train control system based on analyzing log files. Furthermore, to compare the performance and the preference of the Natural Language Generation system, when using two different representations for the train control system in the Data Analysis/Content Determination Task.

The research is completed in 4 stages, which are explained in the following 4 subsections.

6.1.2. Exploratory stage

The research began with getting familiar with the related work on the topic. Papers from the field of Natural Language Generation and anomaly detection were studied and analyzed to get a solid understanding of up-to-date research. Next, the train control system was studied and understood in detailed.

Once the domain knowledge was acquired, two graph representations of the train control system were designed. One representation used more details to represent the system and get more information (states and attributes) from the data (log files). The second representation used less information (states only).

To design the Natural Language Generation system and to perform a requirement analysis, 2 interviews with the software developers of the train control system were conducted. Based on the interview the functional and not functional requirements were set as follows :

- *Functional requirements*: whenever an anomaly is detected in the log files, the Natural Language Generation system should present the user with a summary explaining the anomaly detected
- *Non-functional requirements*: The summary should always include the exact time when the anomalous message was logged. Furthermore, the exact route request specification should be presented to the user. The anomaly should be explained in a textual form in details, including the exact state names where the anomaly was detected. Lastly, a table with all the log messages that are concerned with the route request need to be presented to the user. The table needs to include the cheksum found in the log file, the state of each message and all the attributes with an indication if they were active/non-active at the moment the message was logged.

6.1.3. Implementation stage

Once the requirements were known, the implementation started. The design of the system was established after which the pre-processing was performed. For the data analysis step, the log files were cleaned and messages from the same route request were mapped and stored in a csv file. The two representation of the train control system were hard-coded into state machines which gave each route request a label which showed if the request was anomalous. If the route was indeed anomalous, further information were retrieved from the log file and the state where the anomaly occurred was specified.

Based on the data analysis, the data interpretation, document planning and the microplanning & realizations steps were performed. A template was designed in a form of a syntax tree for each anomaly type. The templates were realized using simpleNLG in Java. The output of the realization were the human readable reports generated by the Natural Language Generation system designed.

6.1.4. Evaluation - Case study

The evaluation of the system was done in a form of a within-subject case study. We designed the case study by specifying the dependent and independent variables of the experiment.

The *independent variables* were the cases chosen to be presented in the reports. The choice was made in such a way that each anomaly type would appear at least once in the reports. Additionally, each participant was presented with two reports. One generated based on the more detailed representation of the train control system, and the other based on the less detailed representation of the train control system.

The *dependent variables* were the order in which each participant was presented with the two reports. Furthermore, to reduce biases, the Latin square model was used to change the order in which each case was presented in each of the reports. This resulted in unique reports for each participant.

Additionally, before the case study was performed, a consent form explaining the case study and how the information gathered will be used was designed and approved by the TU Delft Ethics committee.

Lastly, the questions asked during the case study were establish such that they are suitable for the task performance and human rating evaluation of the system.

6.1.5. Results

The information gathered from the case study were analyzed and the hypothesis were tested. The data was analyzed by using descriptive statistics and a Wilcoxon Sign Tank test. Based on the hypothesis testing the research question was answered. The results and the answer of the research question is discussed in the next subsection (Section 6.2).

6.2. Conclusion

In this research we designed, implemented and evaluated a Natural Language Generation system for the rail domain.

The research question, *When reporting anomalies of a train control system, what degree of detail from the log files should the NLG system use such that the user gets a full understanding of the anomaly?* was answered based on the results of the two hypothesis below.

- H1: The reports generated by the NLG system that uses the more detailed representation of the train control system is more understandable compared to the reports generated by NLG system that uses the less detailed representation of the train control system
- H2: The reports generated by the NLG system that uses the more detailed representation of the train control system is preferred by the domain experts compared to the reports generated by the NLG system that uses the less detailed representation of the train control system

Hypothesis 1 was **rejected** as no statistically significant difference was seen when comparing the performance of the two reports overall. Moreover, we conclude that the performance of both reports was very high which shows that both reports presented the user with a solid explanation of the anomaly detected. Even thought the overall performance of the two reports was not significantly different, we found a significant difference for case 3 - undefined anomaly, that showed a better performance for the report that was generated based on the more detailed train control system representation. Due to the small sample size which is further discussed in the Limitations section, we suggest further analysis and re-testing of this hypothesis.

Hypothesis 2 was **accepted**. The results from the human rating evaluation showed that the participants did prefer the explanations from the report that was generated by using the more detailed representation of the train control system. From the results, it can be seen that the readability measure got the lowest score compared to the accuracy and usefulness measure. By analysing the open questions answers where the participants explained their choices, we saw a trend that suggested that the readability got a lower score from participants that did not have a deep knowledge of the log files. Additionally, most of the participants answered that they were not familiar with seeing the attributes next to the state name, and this new representation would make the explanations more difficult to read. We assume that once the participants get familiarized with this representation, this will not be the case.

Answering the research question

Based on the results we firstly conclude that both reports presented the user with a solid explanation of the anomalies. Furthermore, based on hypothesis 2 we conclude that participants preferred the reports generated by the Natural Language Generation system that uses a more detailed representation of the train control system. Therefore, we conclude that using both the state names and state attributes from the log files, gives a better understanding of the anomalies.

6.3. Lamination

This research had a big challenge with the evaluation of the Natural Language Generation system. That was due to the limitation of the small sample size of domain experts that could take part in the case study. As explained in the Chapter 5, the number people who were part of the case study was 16. Furthermore, after cleaning the data and removing the answers that could not be coded, the analysis of the results was done on 14 participants only. This does not allow us to make strong conclusion. Additionally, if the sample size was bigger, other hypothesis could be tested. For example, we could further divide the domain expert in classes of how high their expertise is. As presented in the results section, based on the open answer question it was noticed that project mangers and project owners (who have less software development experience in the train control system) found the less detailed report easier to read due to the fact that they were not used to seeing the attributes as part of the states. Having the opportunity to evaluate the reports with a bigger sample size, would make it possible to make additional conclusions about the NLG system.

6.4. Future Work

For future research and development on the Natural Language Generation systems in the rail domain we suggest modifying and testing some of the following aspects:

- Using a different anomaly detection technique for the data analysis/content determination task
- Instead of hard coding the state machine, we suggest other researchers to train the state machine using machine learning techniques

Furthermore, it has been explained that this research focuses on one component of the train control system. In the future, there will be added benefit if the other components are further related and other types of log files are used. This will contribute to a more detailed reports and it will make it possible to further spot the exact anomaly. For example, when an anomaly Type 1 is detected (signaling that one or more elements were not able to be set in the requested position), looking at the Element component of the train control system and making a relation with the Rijweg component, would give the

opportunity to point out exactly which element caused the problem.

Lastly, we mentioned that for Case 4 presented in the reports the performance score was low compared to the other 3 cases. This was because some of the participants did not agree that case 4 was an anomaly. As explained earlier, case 4 did not necessarily indicate a harsh anomaly, but maybe a loss in connection between the train control system layers or the components. Another explanation would be that the log file size limit was reached and some messages of the route request were logged in another log file. For this case, we suggest to change the explanation in the NLG report as a warning rather than an anomaly.

Bibliography

[1] SimpleNLG Java. https://github.com/simplenlg/simplenlg. (????). Accessed: 2010-09-30.

[2] Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. Graph based anomaly detection and description: a survey. *Data mining and knowledge discovery* 29, 3 (2015), 626–688.

[3] Lindsay Victoria Allen. 2010. Verification and Anomaly Detection for Event-Based Control of Manufacturing Systems. (2010).

[4] Hadi Banaee, Mobyen Uddin Ahmed, and Amy Loutfi. 2013. Towards NLG for Physiological Data Monitoringwith Body Area Networks. In *14th European Workshop on Natural Language Generation, Sofia, Bulgaria, August 8-9, 2013*. 193–197.

[5] Horst Bunke. 1999. Error correcting graph matching: On the influence of the underlying cost function. *IEEE transactions on pattern analysis and machine intelligence* 21, 9 (1999), 917–922.

[6] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41, 3 (2009), 15.

[7] David L Chen and Raymond J Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*. ACM, 128–135.

[8] José Coch. 1998. SYSTEM DEMONSTRATION INTERACTIVE GENERATION AND KNOWLEDGE ADMINISTRATION IN MULTIMETEO. *Natural Language Generation* (1998).

[9] Mirtha-Lina Fernández and Gabriel Valiente. 2001. A graph distance metric combining maximum common subgraph and minimum common supergraph. *Pattern Recognition Letters* 22, 6-7 (2001), 753–758.

[10] Albert Gatt and Emiel Krahmer. 2018. Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research* 61 (2018), 65–170.

[11] James Hunter, Yvonne Freer, Albert Gatt, Ehud Reiter, Somayajulu Sripada, and Cindy Sykes. 2012. Automatic generation of natural language nursing shift summaries in neonatal intensive care: BT-Nurse. *Artificial intelligence in medicine* 56, 3 (2012), 157–172.

[12] Richard Kittredge, Myunghee Kim, Eli Goldberg, and Alain Polguere. 1994. Sublanguage engineering in the FOG system. In *Proceedings of the fourth conference on Applied natural language processing*. Association for Computational Linguistics, 215–216.

[13] Saad Mahamood and Ehud Reiter. 2011. Generating affective natural language for parents of neonatal infants. In *Proceedings of the 13th European Workshop on Natural Language Generation*. Association for Computational Linguistics, 12–21.

[14] Alexander Maier, Asmir Vodencarevic, Oliver Niggemann, Roman Just, and Michael Jaeger. 2011. Anomaly detection in production plants using timed automata. In *8th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*. 363–369.

[15] Christoph C Michael and Anup Ghosh. 2000. Two state-based approaches to program-based anomaly detection. In *Proceedings 16th Annual Computer Security Applications Conference (ACSAC'00)*. IEEE, 21–30.

[16] Ruslan Mitkov. 1991. *Generating public weather reports*. Universiti Sains Malaysia.

[17] Marcello Pelillo. 1999. Replicator equations, maximal cliques, and graph isomorphism. In *Advances in Neural Information Processing Systems*. 550–556.

[18] Marcello Pelillo. 2002. Matching free trees, maximal cliques, and monotone game dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, 11 (2002), 1535–1541.

[19] François Portet, Ehud Reiter, Albert Gatt, Jim Hunter, Somayajulu Sripada, Yvonne Freer, and Cindy Sykes. 2009. Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence* 173, 7-8 (2009), 789–816.

[20] François Portet, Ehud Reiter, Albert Gatt, Jim Hunter, Somayajulu Sripada, Yvonne Freer, and Cindy Sykes. 2009. Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence* 173, 7-8 (2009), 789–816.

[21] Ehud Reiter. 2007. An architecture for data-to-text systems. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*. Association for Computational Linguistics, 97–104.

[22] Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering* 3, 1 (1997), 57–87.

[23] Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge university press.

[24] Debajit Sensarma and Samar Sen Sarma. 2015. A survey on different graph based anomaly detection techniques. *Indian Journal of Science and Technology* 8, 31 (2015).

[25] Bengt Sigurd, Caroline Willners, Mats Eeg-Olofsson, and Christer Johansson. 1992. Deep comprehension, generation and translation of weather forecasts (Weathra). In *Proceedings of the 14th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, 749–755.

[26] Somayajulu Sripada, Ehud Reiter, and Ian Davy. 2003. SumTime-Mousam: Configurable marine weather forecast generator. *Expert Update* 6, 3 (2003), 4–10.

[27] Somayajulu G Sripada and Feng Gao. 2007. Linguistic Interpretations of Scuba Dive Computer Data. In *Information Visualization, 2007. IV'07. 11th International Conference*. IEEE, 436–441.

[28] Somayajulu G Sripada and Feng Gao. 2007. Summarizing dive computer data: A case study in integrating textual and graphical presentations of numerical data. In *MOG 2007 Workshop on Multimodal Output Generation*. 149.

[29] Mariët Theune, Esther Klabbers, Jan-Roelof de Pijper, Emiel Krahmer, and Jan Odijk. 2001. From data to speech: a general approach. *Natural Language Engineering* 7, 1 (2001), 47–86.

[30] J Treurniet. 2005. *A Finite State Machine Algorithm for Detecting TCP Anomalies*. Technical Report. Technical report, Defence R&D Canada.

[31] Julian R Ullmann. 1976. An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)* 23, 1 (1976), 31–42.

[32] Marian Van Der Meulen, Robert H Logie, Yvonne Freer, Cindy Sykes, Neil McIntosh, and Jim Hunter. 2010. When a graph is poorer than 100 words: A comparison of computerised natural language generation, human generated descriptions and graphical displays in neonatal intensive care. *Applied Cognitive Psychology: The Official Journal of the Society for Applied Research in Memory and Cognition* 24, 1 (2010), 77–89.

A

Appendix A

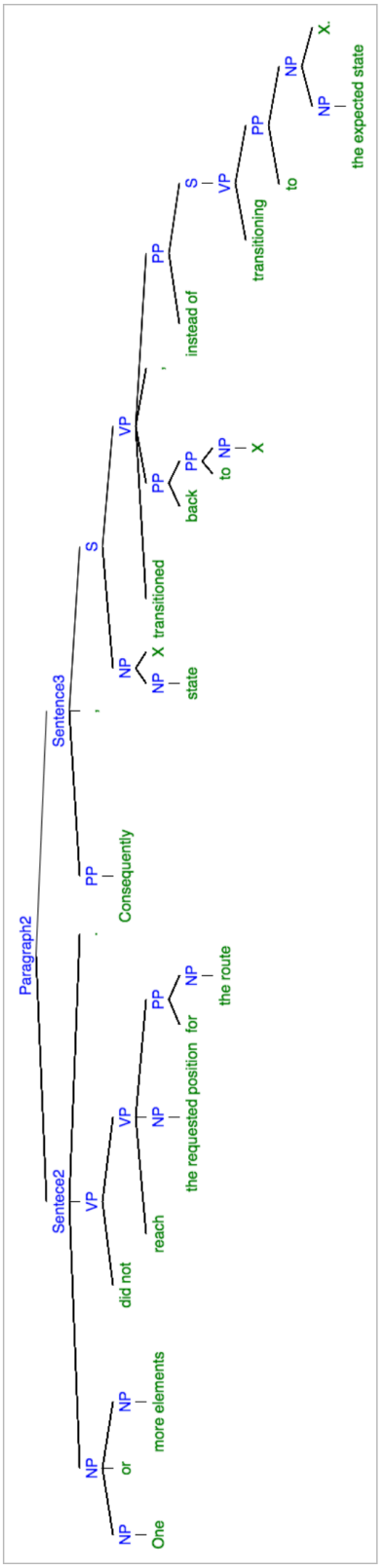

Figure A.1: Syntax tree used for realization of paragraph 2 when explaining Type 1 anomaly with NLG system that uses (*SM + A*)

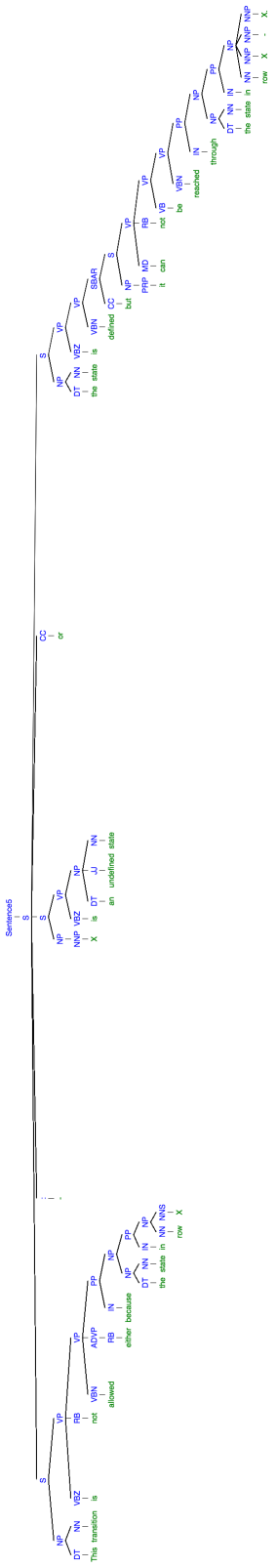

Figure A.2: Syntax tree used for realization of sentence 5 for explaining anomaly type 5(undefined anomaly) for both NLG systems

THESIS EVALUATION

Report 1

Unusual behavior was detected when setting the route 62:76:LLRR.

One or more elements did not reach the requested position for the route. Consequently, state 'Wordt_voorbereid [IR]' transitioned back to 'Gereserveerd [IR]', instead of transitioning to the expected state 'Voorbereid [IR]'.

The table below shows all the states visited and the attributes that were active in the particular state (1 - Actief, 0 - NonActief). Additionally, the time when a state was visited is indicated in the third column. The unusual state transition is the transition from row 4 to row 5.

Row	Checksum	Seconden	Microseconden	RijwegToestand	AU	RR	VR	IR	H	RVR
1	701095711	1554738769	979152	Rust	0	0	0	0	0	0
2	2780274026	1554738772	690953	Rust	0	1	1	1	0	0
3	923368117	1554738772	719985	Gereserveerd	0	0	1	1	0	0
4	3584276470	1554738772	742675	Wordt_voorbereid	0	0	0	1	0	0
5	2289598487	1554738774	204332	Gereserveerd	0	0	0	1	0	0
6	2157779888	1554738776	353584	Gereserveerd	0	0	0	1	1	0
7	2163222222	1554738776	391240	Rust	0	0	0	0	0	0

1. Which unusual behavior occurred?
2. Please mark in the table the messages that correspond to the explained unusual behavior?

Unusual behavior was detected when setting the route 62:76:LLRR.

BEVNL denied an 'InstellenRijweg' request sent by ASTRIS. Therefore, ASTRIS transitioned from state 'Instelopdracht_verstuurd []' to 'Instelopdracht_afgekeurd []', instead of transitioning to state 'Wordt_ingesteld []'.

The table below shows all the states visited and the attributes that were active in the particular state (1 - Actief, 0 - NonActief). Additionally, the time when a state was visited is indicated in the third column. The unusual state transition is the transition from row 6 to row 7.

Row	Checksum	Seconden	Microseconden	RijwegToestand	AU	RR	VR	IR	H	RVR
1	521752591	1554738700	504901	Rust	0	0	0	0	0	0
2	2346702536	1554738729	202667	Rust	0	1	1	1	0	0
3	3615577721	1554738729	232141	Gereserveerd	0	0	1	1	0	0
4	555429962	1554738729	257274	Wordt_voorbereid	0	0	0	1	0	0
5	692074719	1554738731	372933	Voorbereid	0	0	0	1	0	0
6	3423643452	1554738731	442913	Instelopdracht_verstuurd	0	0	0	0	0	0
7	3378695803	1554738731	497998	Instelopdracht_afgekeurd	0	0	0	0	0	0
8	4143216090	1554738739	891437	Instelopdracht_afgekeurd	0	0	0	0	0	0
9	3879276780	1554738739	970883	Rust	0	0	0	0	0	0

1. Which unusual behavior occurred?
2. Please mark in the table the messages that correspond to the explained unusual behavior?

Figure A.3: Example Report 1

Unusual behavior was detected when setting the route 62:76:LLRR.

An undefined state transition occurred. State 'Rust[]' transitioned to state 'Rust[]'. This transition is not allowed either because the state in row 2 - Rust[] is an undefined state or the state is defined but it cannot be reached through the state in row 1 - Rust[].

The table below shows all the states visited and the attributes that were active in the particular state (1 - Actief, 0 - NonActief). Additionally, the time when a state was visited is indicated in the third column. The unusual state transition is the transition from row 1 to row 2.

Row	Checksum	Seconden	Microseconden	RijwegToestand	AU	RR	VR	IR	H	RVR
1	3872026191	1554738756	145313	Rust	0	0	0	0	0	0
2	1437199775	1554738761	537379	Rust	0	0	0	0	0	0

1. Which unusual behavior occurred?
2. Please mark in the table the messages that correspond to the explained unusual behavior?

Unusual behavior was detected when setting the route 62:76:LLRR.

An undefined state transition occurred. State 'Instelopdracht_verstuurd [AU]' transitioned to state 'Instelopdracht_afgekeurd[AU]'. This transition is not allowed either because the state in row 7 - Instelopdracht_verstuurd [AU] is an undefined state or the state is defined but it cannot be reached through the state in row 6 - Instelopdracht_afgekeurd[AU].

The table below shows all the states visited and the attributes that were active in a particular state (1 - Actief, 0 - NonActief). Additionally, the time when a state was visited is indicated in the third column. The unusual state transition is the transition from row 6 to row 7.

Row	Checksum	Seconden	Microseconden	RijwegToestand	AU	RR	VR	IR	H	RVR
1	1437199775	1554738761	537379	Rust	0	0	0	0	0	0
2	4148145999	1554738766	620695	Rust	1	1	1	1	0	0
3	314354231	1554738766	652771	Gereserveerd	1	0	1	1	0	0
4	1072677159	1554738766	870286	Wordt_voorbereid	1	0	0	1	0	0
5	4134918267	1554738768	468936	Voorbereid	1	0	0	1	0	0
6	3406344720	1554738768	524997	Instelopdracht_verstuurd	1	0	0	0	0	0
7	1006143372	1554738768	573681	Instelopdracht_afgekeurd	1	0	0	0	0	0
8	411887338	1554738769	84454	Instelopdracht_afgekeurd	1	0	0	0	1	0
9	3557311504	1554738769	126824	Rust	0	0	0	0	0	0

1. Which unusual behavior occurred?
2. Please mark in the table the messages that correspond to the explained unusual behavior?

Unusual behavior was detected when setting the route 162:118X:RRRR.

This system did not go back to state 'Rust'. This might be because the log file did not contain all the messages related to this 'InstellenRijweg' request.

The table below shows all the states visited and the attributes that were active in a particular state (1 - Actief, 0 - NonActief). Additionally, the time when a state was visited is indicated in the third column.

Figure A.3: Example Report 1 (cont.)

Row	Checksum	Seconden	Microseconden	RijwegToestand	AU	RR	VR	IR	H	RVR
1	693896927	1530537993	85319	Rust	0	0	0	0	0	0
2	1297160597	1530538037	953844	Rust	0	1	1	1	0	0
3	3234177802	1530538037	969004	Gereserveerd	0	0	1	1	0	0
4	3665493062	1530538037	976508	Voorbereid	0	0	0	1	0	0
5	1567508878	1530538038	19487	Instelopdracht_verstuurd	0	0	0	0	0	0
6	1940122011	1530538038	54064	Wordt_ingesteld	0	0	0	0	0	0

1. Which unusual behavior occurred?
2. Please mark in the table the messages that correspond to the explained unusual behavior?

Figure A.3: Example Report 1 (cont.)

THESIS EVALUATION
Report 2

Unusual behavior was detected when setting the route 62:76:LLRR.

One or more elements did not reach the requested position for the route. Consequently, state 'Wordt_voorbereid transitioned back to 'Gereserveerd', instead of transitioning to the expected state 'Voorbereid'.

The table below shows all the states visited and the attributes that were active in a particular state (1 - Actief, 0 - NonActief). Additionally, the time when a state was visited is indicated in the third column. The unusual state transition is the transition from row 4 to row 5.

Row	Checksum	Seconden	Microseconden	RijwegToestand	AU	RR	VR	IR	H	RVR
1	701095711	1554738769	979152	Rust	0	0	0	0	0	0
2	2780274026	1554738772	690953	Rust	0	1	1	1	0	0
3	923368117	1554738772	719985	Gereserveerd	0	0	1	1	0	0
4	3584276470	1554738772	742675	Wordt_voorbereid	0	0	0	1	0	0
5	2289598487	1554738774	204332	Gereserveerd	0	0	0	1	0	0
6	2157779888	1554738776	353584	Gereserveerd	0	0	0	1	1	0
7	2163222222	1554738776	391240	Rust	0	0	0	0	0	0

1. Which unusual behavior occurred?
2. Please mark in the table the messages that correspond to the explained unusual behavior?

Unusual behavior was detected when setting the route 62:76:LLRR.

BEVNL denied an 'InstellenRijweg' request sent by ASTRIS. Therefore, ASTRIS transitioned from state 'Instelopdracht_verstuurd' to 'Instelopdracht_afgekeurd', instead of transitioning to state 'Wordt_ingesteld'.

The table below shows all the states visited and the attributes that were active in a particular state (1 - Actief, 0 - NonActief). Additionally, the time when a state was visited is indicated in the third column. The unusual state transition is the transition from row 6 to row 7.

Row	Checksum	Seconden	Microseconden	RijwegToestand	AU	RR	VR	IR	H	RVR
1	521752591	1554738700	504901	Rust	0	0	0	0	0	0
2	2346702536	1554738729	202667	Rust	0	1	1	1	0	0
3	3615577721	1554738729	232141	Gereserveerd	0	0	1	1	0	0
4	555429962	1554738729	257274	Wordt_voorbereid	0	0	0	1	0	0
5	692074719	1554738731	372933	Voorbereid	0	0	0	1	0	0
6	3423643452	1554738731	442913	Instelopdracht_verstuurd	0	0	0	0	0	0
7	3378695803	1554738731	497998	Instelopdracht_afgekeurd	0	0	0	0	0	0
8	4143216090	1554738739	891437	Instelopdracht_afgekeurd	0	0	0	0	0	0
9	3879276780	1554738739	970883	Rust	0	0	0	0	0	0

1. Which unusual behavior occurred?
2. Please mark in the table the messages that correspond to the explained unusual behavior?

Figure A.4: Example Report 2

Unusual behavior was detected when setting the route 62:76:LLRR.

BEVNL denied an 'InstellenRijweg' request sent by ASTRIS. Therefore, ASTRIS transitioned from state 'Instelopdracht_verstuurd' to 'Instelopdracht_afgekeurd', instead of transitioning to state 'Wordt_ingesteld'.

The table below shows all the states visited and the attributes that were active in a particular state (1 - Actief, 0 - NonActief). Additionally, the time when a state was visited is indicated in the third column. The unusual state transition is the transition from row 6 to row 7.

Row	Checksum	Seconden	Microseconden	RijwegToestand	AU	RR	VR	IR	H	RVR
1	1437199775	1554738761	537379	Rust	0	0	0	0	0	0
2	4148145999	1554738766	620695	Rust	1	1	1	1	0	0
3	314354231	1554738766	652771	Gereserveerd	1	0	1	1	0	0
4	1072677159	1554738766	870286	Wordt_voorbereid	1	0	0	1	0	0
5	4134918267	1554738768	468936	Voorbereid	1	0	0	1	0	0
6	3406344720	1554738768	524997	Instelopdracht_verstuurd	1	0	0	0	0	0
7	1006143372	1554738768	573681	Instelopdracht_afgekeurd	1	0	0	0	0	0
8	411887338	1554738769	84454	Instelopdracht_afgekeurd	1	0	0	0	1	0
9	3557311504	1554738769	126824	Rust	0	0	0	0	0	0

1. Which unusual behavior occurred?
2. Please mark in the table the messages that correspond to the explained unusual behavior?

Unusual behavior was detected when setting the route 162:118X:RRRR.

This system did not go back to state 'Rust'. This might be because the log file did not contain all the messages related to this 'InstellenRijweg' request.

The table below shows all the states visited and the attributes that were active in a particular state (1 - Actief, 0 - NonActief). Additionally, the time when a state was visited is indicated in the third column.

Row	Checksum	Seconden	Microseconden	RijwegToestand	AU	RR	VR	IR	H	RVR
1	693896927	1530537993	85319	Rust	0	0	0	0	0	0
2	1297160597	1530538037	953844	Rust	0	1	1	1	0	0
3	3234177802	1530538037	969004	Gereserveerd	0	0	1	1	0	0
4	3665493062	1530538037	976508	Voorbereid	0	0	0	1	0	0
5	1567508878	1530538038	19487	Instelopdracht_verstuurd	0	0	0	0	0	0
6	1940122011	1530538038	54064	Wordt_ingesteld	0	0	0	0	0	0

1. Which unusual behavior occurred?
2. Please mark in the table the messages that correspond to the explained unusual behavior?

Figure A.4: Example Report 2 (cont.)

2018-07-02T13:26:34.4012Z|IDCR_ZL01|niemand|Astris08|EMSCorrelationID='geen-id'EMSBericht='<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope" xmlns="ESBConsumerEnvelope" xmlns:b="http://docs.oasis-open.org/wsn/b-2" xmlns:esb="ESBConsumerWSDL" xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsrf-bf="http://docs.oasis-open.org/wsrf/bf-2">
<soapenv:Header>
<wsa:To>niemand</wsa:To>
<wsa:From>
<wsa:Address>ABS</wsa:Address>
</wsa:From>
<wsa:ReplyTo>
<wsa:Address>ABS</wsa:Address>
</wsa:ReplyTo>
<wsa:MessageID>1530537994401214</wsa:MessageID>
<wsa:Action>NotifyConsumer</wsa:Action>
</soapenv:Header>
<soapenv:Body>
<b:Notify>
<b:NotificationMessage>
<b:SubscriptionReference>
<wsa:Address>geen-ref</wsa:Address>
</b:SubscriptionReference>
<b:Topic Dialect="Simple">
<esb:topicName>Rijweg</esb:topicName>
</b:Topic>
<b:ProducerReference>
<wsa:Address>ASTRIS-IDCR</wsa:Address>
</b:ProducerReference>
<b:Message>
<esb:msgContainer>
<esb:msgHeader>
<esb:berichtVolgnr>13</esb:berichtVolgnr>
<esb:volgendeIsSnapshot>false</esb:volgendeIsSnapshot>
</esb:msgHeader>
<esb:msgBody>
<esb:ID>SEIN:110:PPLG:ZL:1:SEIN:86:PPLG:ZL:1:LL</esb:ID>
<RIM:Notification xmlns:RIM="RailinfraMelden" xmlns:RIT="RailinfraTypes">
 <RIM:wijzigingRijweg_1>
  <RIT:SafetyCode>
   <RIT:Checksum>2705496246</RIT:Checksum>
   <RIT:Volgnummer>0</RIT:Volgnummer>
   <RIT:Tijdstempel>
    <RIT:Seconden>1530537993</RIT:Seconden>
    <RIT:Microseconden>50473</RIT:Microseconden>
   </RIT:Tijdstempel>
  </RIT:SafetyCode>
  <RIT:RijwegIdentificatie>

```
    <RIT:BeginIdentificatie>
     <RIT:ObjectType>SEIN</RIT:ObjectType>
     <RIT:ObjectNaam>110</RIT:ObjectNaam>
     <RIT:GebiedType>PPLG</RIT:GebiedType>
     <RIT:GebiedNaam>ZL</RIT:GebiedNaam>
     <RIT:InfraVersie>1</RIT:InfraVersie>
    </RIT:BeginIdentificatie>
    <RIT:EindIdentificatie>
     <RIT:ObjectType>SEIN</RIT:ObjectType>
     <RIT:ObjectNaam>86</RIT:ObjectNaam>
     <RIT:GebiedType>PPLG</RIT:GebiedType>
     <RIT:GebiedNaam>ZL</RIT:GebiedNaam>
     <RIT:InfraVersie>1</RIT:InfraVersie>
    </RIT:EindIdentificatie>
    <RIT:LRString>LL</RIT:LRString>
   </RIT:RijwegIdentificatie>
   <RIT:RijwegToestand>
    <RIT:ToestandRijweg>Rust</RIT:ToestandRijweg>
    <RIT:AttributenRijweg>
     <RIT:RWA_AnnulerenRijweg>NietActief</RIT:RWA_AnnulerenRijweg>
     <RIT:RWA_AnnulerenSTSRoute>NietActief</RIT:RWA_AnnulerenSTSRoute>
     <RIT:RWA_Automaat>NietActief</RIT:RWA_Automaat>
     <RIT:RWA_Herroepen>NietActief</RIT:RWA_Herroepen>
     <RIT:RWA_InstellenRijweg>NietActief</RIT:RWA_InstellenRijweg>
     <RIT:RWA_ReserverenRijweg>NietActief</RIT:RWA_ReserverenRijweg>
     <RIT:RWA_RijwegVrijrijden>NietActief</RIT:RWA_RijwegVrijrijden>
     <RIT:RWA_STSRoute>NietActief</RIT:RWA_STSRoute>
     <RIT:RWA_VoorbereidenRijweg>NietActief</RIT:RWA_VoorbereidenRijweg>
     <RIT:RWA_EmplacementStopdoor>NietActief</RIT:RWA_EmplacementStopdoor>
     <RIT:RWA_GoederenCriterium>NietActief</RIT:RWA_GoederenCriterium>
     <RIT:RWA_VrijebaanStopdoor1>NietActief</RIT:RWA_VrijebaanStopdoor1>
     <RIT:RWA_VrijebaanStopdoor2>NietActief</RIT:RWA_VrijebaanStopdoor2>
     <RIT:RWA_VrijebaanStopdoor3>NietActief</RIT:RWA_VrijebaanStopdoor3>
     <RIT:RWA_VrijebaanStopdoor4>NietActief</RIT:RWA_VrijebaanStopdoor4>
    </RIT:AttributenRijweg>
   </RIT:RijwegToestand>
  </RIM:wijzigingRijweg_1>
</RIM:Notification>
</esb:msgBody>
</esb:msgContainer>
</b:Message>
</b:NotificationMessage>
</b:Notify>
</soapenv:Body>
</soapenv:Envelope>'
```


Figure A.5: Example message from an IDCR log file