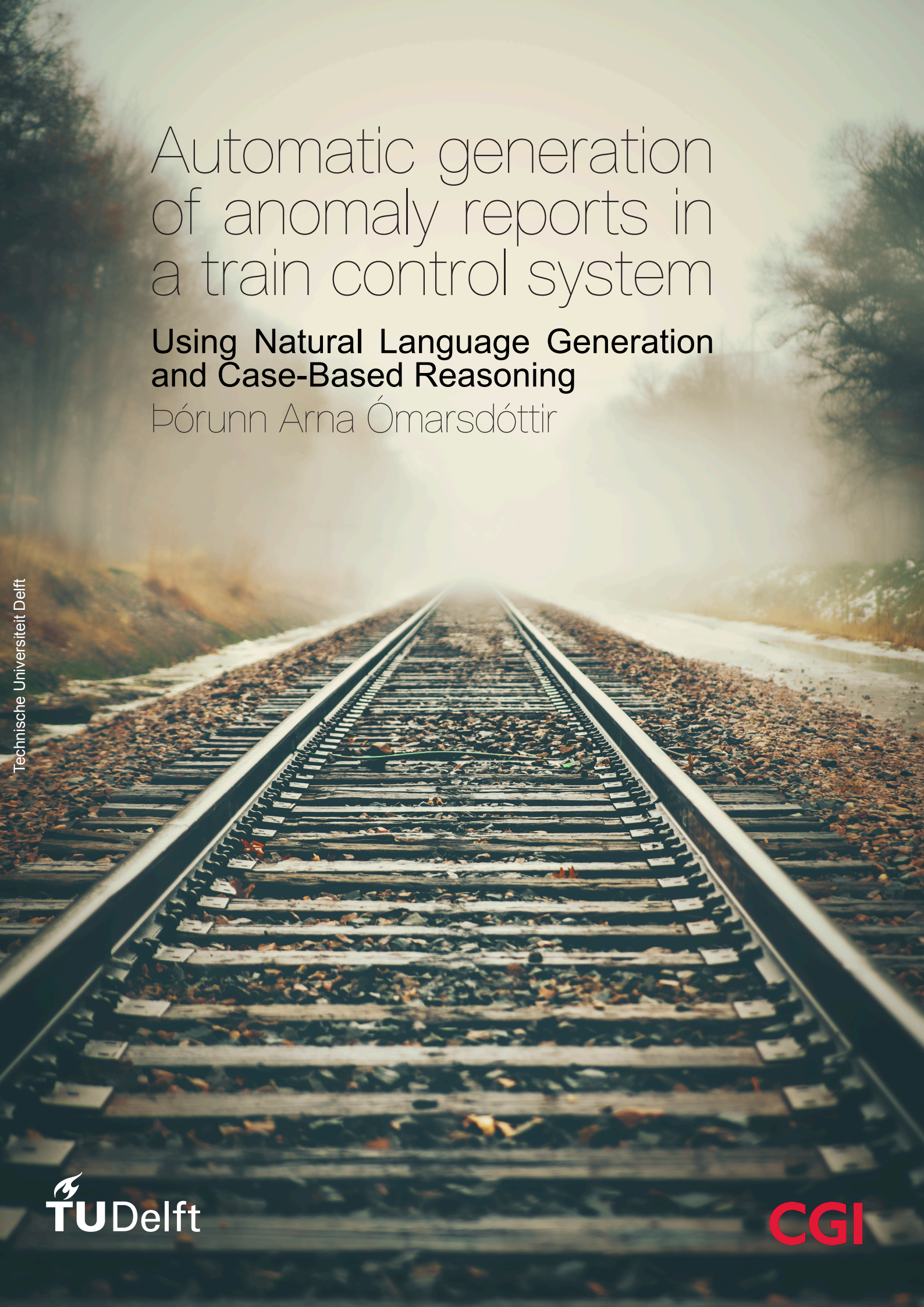
Automatic generation of anomaly reports in a train control system

Using Natural Language Generation and Case-Based Reasoning

Þórunn Arna Ómarsdóttir

Technische Universiteit Delft



TU Delft

CGI

Automatic generation of anomaly reports in a train control system

Using Natural Language Generation and Case-Based Reasoning

by

Þórunn Arna Ómarsdóttir

In partial fulfillment of the requirements to obtain the degree of

Master of Science
Computer Science
Data Science and Technology Track

from the Delft University of Technology.
To be defended publicly September 15th, 2020 at 14:00.

Student number:	4917499	
Project duration:	December 1, 2019 – September 15, 2020	
Thesis committee:	Prof. dr. ir. G. J. P. M. Houben,	TU Delft, chair
	Dr. N. Tintarev,	TU Delft, supervisor
	Dr. M. T. J. Spaan,	TU Delft
	S. Najafian,	TU Delft, daily supervisor

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft Delft University of Technology

Abstract

In this thesis, we study automatically generating explanatory reports for anomalous incidents in a train control system (TCS) using Natural Language Generation (NLG). A TCS is a type of safety-critical software that allows train controllers to correctly set the tracks for a train to pass. The goal of this research is to process the majority of log files generated by such a system, detect anomalies that have occurred and represent that data in human-readable reports that explain anomalous incidents.

The reports are generated by making use of NLG techniques, namely data-to-text. We design an NLG pipeline that incorporates novel graphical components. We perform all steps of the report generation process, which is the processing of data, anomaly detection and representing the data in natural language. To process the data we incorporate complex domain-specific rules which require extensive work to accommodate five separate types of event log files. Analyzing logs and detecting anomalous incidents proved complex due to the complicated structure of logs and the ill-defined relationships between different log file types. The log data explained with the NLG pipeline is more extensive than we have seen in literature, as it needs to be linked using complex domain-specific rules and includes temporal and geographical aspects of the railway system. Furthermore, the detected anomalies are very diverse and thus it is unclear what aspects should be included in a report for each anomalous case. We research whether a purely textual presentation or a combination of modalities provides higher information quality.

The product of the NLG system are three presentation versions of incident reports, *Text*, *TimeText* and *RailText*. Each version focuses on different important aspects of the nature of the data, to explain anomalous incidents. Due to the diverse properties of the anomalous cases detected, we implement a Case-Based Reasoning (CBR) system to predict the appropriate presentation to explain each incident. We evaluate our work in two expert-based evaluation phases. The first phase evaluates the quality of the different report presentations. The second phase evaluates the feasibility of the CBR system and the quality of the reports chosen by CBR.

The work in this thesis is significant as we manage to translate complex data into human-readable reports that are well received by experts and help in understanding anomalous cases. We learn that there is a difference in what presentation format provides the highest information quality of reports depending on the anomalous case being explained. Therefore we conclude that the properties of anomalous cases should be taken into account when generating anomalous reports in a TCS to ensure the highest perceived information quality of the reports. CBR is shown to perform that task well. Furthermore, we find that data familiarity of experts effects their preferences for report presentations.

Key words: Natural Language Generation, Multi-modality, Log mining, Anomaly detection, Anomaly explanation, Case-Based Reasoning, Railway domain.

Preface

Working on this thesis has been very informative and has allowed me to grow as a computer scientist and researcher. It is has been a great opportunity to get familiar with the industry and the research community at the same time. This combination provoked challenges since it generates the need to contribute a project that is helpful and significant for the company while being academically interesting. In retrospect, the choice of domain is rather ironic for me, since I come from a country which does not have any train system and I do not speak the language in which the data is represented. This research project has proven challenging, as the first months were spent purely on studying the data and identifying the problem in place. The direction of the project took many changes in the first months, but ended with a scope I am very pleased with.

I am very thankful for the guidance from all of my supervisor team. Vincent van der Goes has been supportive and made sure I had all the domain information and connection to the system experts I needed for a successful project. The same can be said for Rutger Heunks, the manager of the ASTRIS team at CGI. Shabnam Najafian has been a very helpful daily supervisor, always available for assisting me when I most needed it. Dr. Nava Tintarev has provided useful feedback that has helped me formulate and scope the thesis.

I would also like to thank the domain experts of the train control system here in the Netherlands, which are the evaluators of the project. Without them I would not have been able to evaluate the system created.

I want to thank my family and friends for supporting me when I needed support, encouraging me to keep going when I needed encouragement and always being there for me. Even though we have been apart for the most part of the last two years I know I can always count on you. Lastly, I would like to thank Tómas, my partner. He has been my rock through the span of this thesis, given me hugs, feedback and ideas which have helped me shape the thesis into its final form. For that, I am so grateful.

Þórunn Arna Ómarsdóttir
Delft, September 2020

Contents

List of Figures

List of Tables

Acronyms

CBR	Case Based Reasoning.
D2T	Data-to-text.
KI	Knowledge-Intensive.
KL	Knowledge-light.
NLG	Natural Language Generation.
NLP	Natural Language Processing.
T2T	Text-to-text.
TCS	Train Control System.

Terms & Names

ARC	Log file type of interest for the system. Contains information on all requests that are given to ASTRIS.
ASTRIS	Train control system layer maintained by CGI and the system of interested for this thesis.
BevNL	Log file type of interest for the system. Contains information on all communication between ASTRIS and the underlying safety layer.
IDCE	Log file type of interest for the system. Contains information on all element related requests, responses and state updates.
IDCR	Log file type of interest for the system. Contains information on all route related requests, responses and state updates.
Meldingen	Log file type of interest for the system. Contains information on all errors and alerts produced by the system.
NPV	Will be used over a Dutch railway term, “Niet profiel vrij situatie”. Which means that there is a possibility for collision of two trains within the dedicated area..
RailText	Presentation version containing text and graphical components showing railway overview.
Text	Presentation version only containing texts on sentence and table format.
TimeText	Presentation version containing text and graphical components showing data entries on a timeline.

1 Introduction

Generating human-readable data summaries of complex data is an important and difficult task. Some research effort has gone into this task in various domains. This is most often done with the help of Natural Language Generation (NLG) methodologies. To name some relevant recent examples we have [60] where soccer summaries are generated from match data and [55] where tweets are generated from traffic data. Furthermore, the effects of enhancing such reports with graphical data components by making them multi-modal have been of interest to researchers. Graphs where generated in [50] and their addition is shown to be useful to interpret textual description as well as in [29] where graphical and textual summaries are compared for interpreting medical data.

In this thesis, we research using NLG to automatically generate reports that are augmented with multimodal data. We are presented with a real-world problem within the railway domain by CGI. Which we attempt to solve, by designing and applying our methodologies to the railway domain. Our work consists of detecting anomalous incidents in a complex system, using techniques that are inspired by state-of-the-art literature. To explain the incidents we provide methodologies to automatically generate reports using NLG, along with graphical representations of data. The graphical components is included to further illustrate the temporal and geographical aspects of the railway system. We aim to account for the variability of anomalies that can occur by presenting a Case-based Reasoning system that predicts the most suitable modality combination for each anomalous case.

In this introductory chapter, we start by motivating our work in Section 1.1. In Section 1.2, we present our research questions and sub-questions. Next, we highlight the contributions of the thesis in Section 1.3 and lastly present the outline for the rest of this thesis project.

1.1. Research Motivation

A Train Control System (TCS) is a complex and safety-critical system. Such systems require good infrastructure to be able to be maintained efficiently. It is a very important quality for the management of a safety-critical system to be able to analyze the data generated by the system efficiently so that it is possible to detect and analyze faults and failures. NLG systems have been shown to be useful to generate precise and useful data summaries in various domains for data analysis and to assist decision-making. NLG has been applied to a good effect to various domains. There are still multiple domains yet to explore though. The domain of railway infrastructure has not yet been researched in terms of NLG. The only recent work in this specific domain was done by Urumovska [58] in 2019. Her research was comprised of an NLG system applied in the same TCS as we will be researching.

The motivation for applying an NLG based methodology to the domain of interest is a real-world problem. When some incident occurs in the TCS it takes extensive work of an expert of the system to understand what happened. Currently, all analysis of the contents of log files is done manually by experts on the raw log files. The log entries need to be linked using various data mappings since the underlying mechanism in the system have complex relationships. This procedure, where log files are investigated in detail and log entries matched together, is difficult and time-consuming. The logging is complex and stored in many different types of log files. Furthermore the sizes of produced log files each day are enormous. It it therefore not feasible to do this manually as it is currently done.

Suitable methodologies to cover all aspects of this incident analysis need to be found. The main goal of our project is therefore to create a system that can assist those experts to identify anomalies and what caused a problem. This will speed up the handling of safety-critical operations, by automatically generating reports which explain incidents. By adapting current NLG methodologies found in literature, we aim to obtain the advantages NLG systems can deliver. That is, precise, automatically created human-readable reports which are understandable and account for domain rules. By doing this we hope to show the feasibility of using these reports to improve methods to explain incidents in complex domains. Furthermore, by applying our methodology to the railway domain, we hope to show the possibility of applying similar methodologies to other related domains or system types.

In the former research related to the same TCS system, Urumovska [58] researched how detailed reports should be when reporting anomalies within the TCS. Only a small fraction of the possible data generated by the system was in the scope for that project. In contrast to her work, we will be handling a larger and more complex part of the available data. We will be including more data, with more complex relationships, so we will be processing broader and more detailed information. Urumovska's [58] work shows that there is potential to use NLG for a TCS. However, it is still unclear how well NLG can perform with a larger, more complex TCS. If NLG can be utilized to good effect on such a complex system, we can show the feasibility of applying NLG to other complex systems. We will work from the assumption that the finding of the former research is valid, and include more details, rather than less details, in the reports. Due to the increased complexity of the data being handled, we expect that the anomalous incidents that need to be covered will be much more diverse. Thus, we explore including different modalities in the generated reports to provide different perspectives of the data when explaining anomalous incidents.

There are multiple challenges we will be facing in this work. Firstly, we need to be able to model the data correctly so that we can detect anomalies that have occurred in the system and suitably represent them in the reports. The data we are working with in this project is complex and has many properties that need to be handled carefully to be able to match data entries correctly and extract the right parts of data to convey in reports. The matching of data entries across different log entry types is vital to be able to tell a complete story showing process information from all system components of interest. To do that there are temporal and geographical factors that need to be taken into account. Another challenge lays in the content selection for the reports; since the TCS is always quite busy it is a challenge to select the right amount of data, so as not to fill the generated reports with unnecessary data.

The failures that occur within a TCS system can be very diverse and occur for various reasons. Therefore we are interested in finding the most appropriate graphical presentation for the report generated for each incident. That is, we want to be able to pick the report generation template that is most suitable. Case-based reasoning (CBR) has been applied to other template selection problems for NLG [3, 21], but not in a related domain to ours or with the goal of selecting the most suitable presentation format for the generated reports. CBR has proven to be good for dynamic domains where new problems arise frequently and the system needs to be continuously learning, revising and updating. We will investigate the possibility to use CBR technology for the task of selecting a presentation template during the NLG process.

The initial idea for applying these research technologies to the domain of railway infrastructure comes from a research topic suggested by CGI NEDERLAND B.V. The initial problem statement was: "What we(CGI) need is better tooling to be able to understand what has happened in an incident and seeing the connections between the relevant information. The outcome should preferably be a readable report of this problem analysis". After hearing this statement and by getting to know the domain and studying obstacles we ended up with the current research setup of this project.

1.2. Research questions

As has been described above in the research motivation section, the goal of the project is to explain anomalous incidents in a TCS with the generation of human readable reports. More detailed goals can be defined to better explain the overall goal. The smaller, more detailed goals are to:

- design and implement a multimodal NLG system that can select content from log files of a TCS and explain detected anomalies in reports meant for domain experts.
- account for the diverse anomalous incidents that can occur and find the most suitable presentation for each one.

To meet these goals, we will design and implement one model of the TCS that will be used in an NLG pipeline to represent the system. Additionally, we will design two dynamic graphical components that can be inserted as extensions to the NLG pipeline. To explore how the different graphical components influence the performance of the reports, we will generate three different report versions each focusing on a different presentation format. The three presentation versions are:

- *Text*: A report containing only text, formatted into sentences and tables.
- *TimeText*: A report that contains everything from *Text* and graphical components that focus on explaining the temporal nature of the data.
- *RailText*: A report that contains everything from *Text* and graphical components that focus on explaining the geographical nature of the data.

To measure how well we manage to meet our goals we will answer the main research question:

> *When generating anomalous incident reports in a train control system, how should presentation formats be adaptively selected for incidents to provide the highest perceived information quality for domain experts?*

We will answer three research sub-questions to reach a conclusion regarding our main research question.

1. What report presentation performs best in terms of perceived information quality and how does it depend on the properties of the anomalous incident?
2. To what extent do expert preferences of report presentations associate with their expert profiles in terms of data familiarity and work experience?
3. How is Case-Based Reasoning, as an adaptive template selection method for anomalous incidents, where incidents are represented with variables, perceived by experts?

Research sub-question 1 focuses on measuring perceived information quality for different anomalous incident cases. The information quality is evaluated in terms of perceived understanding and perceived preference. There we aim to find out what report presentation yields the highest perceived information quality for each case. The adaptive part of the main research question will be covered in sub-questions 2 and 3. Sub-question 2 focuses on how the findings from sub-question 1 are related to the professional properties of experts. Sub-question 3 focuses on how to adapt the report generation process to account for diverse anomalous incidents. We will attempt to do so by using a CBR approach.

1.3. Contributions

The contribution of this research is a report generation system that makes use of NLG methodologies and multi-modality. The model can be used to generate reports on a technical level to provide domain experts with a high understanding of the incident being explained. Furthermore, the system is extended with a CBR system. The CBR system can select the modality that is considered most suitable by experts for an unseen case that should be explained.

This thesis project aims to show the feasibility of using NLG, annotated graphical components and a CBR system to explain anomalous incidents in a complex system. The application of this system is intended for our domain experts. The anomaly detection is highly domain-specific, but the design of our report generation could be applied to other domains. Therefore, by showing the feasibility of our work we aim to show the possibility of applying a similar methodology to other complex systems. NLG has become increasingly popular in recent years; we aim to apply methodology from relevant literature to our research.

Here, we extend upon a recent thesis project [58], that applied NLG to good effect to a subset of the complex system of interest. This project focuses on extending this work in two specific ways. Firstly, by using a much larger content base that requires different content determination methods. Secondly, by including graphical components to better visualize anomalous incidents and compare their effects.

The content base that we work with is composed of various types of log files. Urumovska [58] based her work on one log file type, while we work with five. By using this amount of log file types, multiple

complex relationships between the data found in the log files are introduced. Thus, we contribute a system that can efficiently handle this complex data and detect anomalies. The gain of our proposed methodology is to greatly simplify the processing of massive amounts of logging data.

Because of the complex nature of the domain and data, we identify diverse anomalous cases. After analyzing some cases we see the potential of introducing graphical components to the reports. The data we process is both temporally and geographically dependent. Thus, we propose two novel automatically generated and annotated graphical components. One component is focused on explaining the temporal nature of the data. The other component explains the geographical nature of the data in terms of the railway network. With the addition of graphical components to reports, we hope to provide higher perceived information quality over purely textual reports.

We research the possibility to apply CBR to our domain, to account for the dynamic nature of the domain. To our knowledge, using CBR for generating explanatory reports has not been done in a similar domain. We contribute a novel CBR system that is inspired by methodologies from literature [3, 21]. This CBR system allows us to dynamically select the presentation templates for each anomalous incident and provides the possibility for expert revision if necessary.

Our work is applied to a real-world system with the help of domain experts. Our results thus show how well our methodology can work on real problems. That motivates the potential of applying our work to any complex system.

1.4. Thesis Outline

The format of the rest of the thesis report is as follows:

Chapter 2: **Background & Related Work**

This chapter is intended to familiarize the reader with existing research that is relevant to the project. We get to know methodologies that have been used for data processing, report generation and evaluation of generated reports. We aim to find which methodologies are most suitable to use in this project. Furthermore, we will point out an identified research gap which we aim to provide insights for

Chapter 3: **Setting of the project**

Here the domain of interest, a real-world problem provided by CGI, is introduced in detail. We provide insights into the importance, challenges and reasons for tackling the problem in this thesis.

Chapter 4: **Requirement Analysis**

This chapter covers the execution and results of a requirement analysis performed. There we get to know the pool of experts and their requirements for the system.

Chapter 5: **Report Generation Methodology**

Chapter 5 introduces and explains all methodologies used in this thesis. We will cover all steps taken when generating explanatory anomaly reports and dynamically choosing modality presentations.

Chapter 6: **Evaluation**

This chapter will explain the evaluation methodologies used to evaluate the competence of the system to meet its goals. Defined hypotheses will be introduced and tested in an expert base user study. Furthermore, the results from the user study will be presented and analyzed.

Chapter 7: **Conclusion, discussion and future work**

Finally, we will conclude our findings by answering our research questions and discussing our results. The known limitations of our work will be identified as well as possible future research work.

2

Background & Related Work

In this chapter we will discuss the tasks that are important for the application of this thesis; the generation of reports explaining diverse anomalies of a TCS to its experts by making use of NLG. We will investigate different techniques and try to find out what are the most applicable methodologies and applications for the project. Based on the findings of the literature review, the methodology for my application and experiments will be chosen. The goal of this chapter is to get familiar with the work that has been done recently in parts of computer science and information technology fields that are relevant to the scope of the problem.

The challenges that we will be facing in this project concern the complexity of the domain of the project and how its data can be formalized in a good manner. It is important for safety-critical systems, like TCS, to be able to communicate their logged data and have supporting systems to analyze it informatively, suitably and efficiently. The log files contain information about direct commands from train controllers and all consequences of a performed command as it goes deeper through the system. By analyzing the log files efficiently the information gained from passed time can be put to good use to understand and fix problems that may be present in the system. These problems can be caused by development defects, not meeting specified requirements or hardware failures.

To perform a linguistic description task for any complex phenomenon there are quite a few aspects that need to be kept in mind. According to Alonso et al. [5], the steps to be taken include: ”Careful analysis of the phenomenon under consideration, regarding the communicative goal, audience background and the set of natural language expressions most commonly used in the context of the application domain”. The design of a report template customized to the accordance of the audience requirements is important. Additionally, it is important to consider the design and implementation of a computational structure that can process raw data, interpret it and produce a report with the most relevant information for the user in each case. This is a good list to keep in mind when reading through this section to understand why we will be looking into multiple research directions.

In this chapter we will get familiarized with all main methodologies used in this thesis to go from raw log files to human readable reports and their evaluation. We will approach this all through an NLG setting and see where all steps we need to take fit into the flow of an NLG system. In Section 2.1 we will get an introduction to NLG and its main properties. Section 2.2 will furthermore explain the main architectures for designing NLG systems and their variations to provide insights in possibilities in design for our system. Section 2.3 introduces NLG applications that relate to the system being designed in this thesis. We point out challenges and important aspects to keep in mind while developing NLG systems. Section 2.4 introduces Case-Based Reasoning and its properties. We learn why it is suitable to apply in our domain setting for prediction presentation versions for unseen incidents. Section 2.5 investigates different ways to process data from log files and how to detect anomalous patterns from them. Section 2.6 covers evaluation methodology for automatically generated text. Finally, we will end with a conclusion of our findings in Section 2.7 and explaining an identified research gap in Section 2.8

2.1. What is Natural Language Generation

NLG has proven to be a very useful technology to translate raw data and keywords into sentence formed text. As we plan to create human readable reports in this thesis, NLG is a very promising methodology to apply. Therefore we will start by getting familiar with the methodology and learn whether it is a suitable technology for us to use.

NLG is the technology of automatically generating text from non-linguistic input. It was initially considered to be a sub-field of Natural Language Processing (NLP) but has developed to just be a closely related research field instead of a descendant of NLP. In a rather old survey paper of the field by Reiter and Dale [41], NLG is characterized as a sub-field of artificial intelligence and computational linguistics, which is still a good representation of the field since the term is as broad as the possibilities of the field itself. In that survey paper, the authors, Reiter and Dale, furthermore say that the goal of NLG systems is to combine knowledge about language and the desired application domain in order to automatically produce documents, reports, explanations, help messages or other kinds of texts. There are endless possibilities in the application of NLG. It is a rather broad and young research field, so many cases are yet to be explored.

NLG is a solution that is suitable for generating text that will not necessarily have a large audience. That is because it is generally less expensive to generate text with the help of an automatic process, rather than get a human writer to compose comparable text. This is because no additional resources are needed to generate each text when the system has already been developed. However, there are deviations from this rule if a system is very expensive to compose and the system is not heavily used [41].

NLG techniques are suitable for a large variety of precise tasks within text generation. Such as accounting for personality, including metaphors, tailoring text to specific audiences and representing knowledge for readers on different knowledge levels. Entertaining and visionary text can also be produced using NLG techniques, but that has not been researched extensively yet. Jokes and narratives can be generated. Furthermore, NLG systems can show affection and style in its conveyed text. These aspects will not be in a particular focus on this current research project since there is not much need for personalization in the project setting at the moment, but could be integrated later if some personalization would be wanted. The list of these different possibilities is meant to briefly show the user the wide range of achievability of NLG systems.

2.1.1. Different properties of NLG systems

There are many different levels of NLG systems, some are simple template filling technologies, others are more complex and can reflect changes and variations specific to a certain domain automatically. According to Saliby [45], the levels of complexity in NLG systems can be split into three categories.

- Firstly, *canned text systems*, where words are copy-pasted systematically to form a text.
- Next there are *template filling systems*, where data is entered into predefined slots.
- Lastly, *advanced NLG systems*, which are flexible and suitable for emerging situations.

In some template filling systems and all advanced systems, choices regarding content selection, lexical selection, sentence- and discourse structure need to be made. According to Hervás and Gervás [21], template-based solutions rely on reusing fragments of text from the relevant domain. Furthermore, they state that templates partly solve the need for having an explicit grammar in NLG reports.

Template systems have the advantage of allowing for full control over the quality of the output and avoiding the occurrences of non grammatical structures in the generated text, according to Gatt and Krahmer [16]. Reiter and Dale [41] and Deemter et al. [13] discuss that template systems can be difficult to maintain and update. They furthermore state that they generate less varied output, which they pose as a disadvantage. Little variability of text is however not always a bad thing since sometimes data should always be consistently expressed, which is the case for the system designed in this thesis. Deemter et al. [13] discuss the difference between template-based and advanced NLG, where they refer to advanced NLG as “Real” NLG. They discuss that templates have a certain advantage over advanced systems since they are able to generate text in cases where good linguistic rules are not available and can be helpful for highly specific conditions. The main conclusion of [13] is that there is no clear line between template based and advanced systems since there are numerous variations of

both types. This is in fact more of a type spectrum instead of the exact types specified here above. Template systems can include variability and be dynamic. Advanced systems can skip certain parts of the NLG pipeline and use shortcuts, which results in a system that is not more adaptable than a dynamic template system.

Adeyanju [3] categorizes techniques for automatic text generation into two categories. They are Knowledge-Intensive (KI) and Knowledge-light (KL) approaches. KI is often also referred to as knowledge-based methods. The KI approaches require extensive consultation with domain experts during the corpus analysis, but the KL approaches rely nearly solely on automated statistical methods. Since the domain we are working is has many domain specific rules that need to be accounted for in the design of an NLG system, the approach we will be designing in this research project should be categorized as KI.

Gatt and Krahmer [16] have composed a thorough survey of the history and current methodology of the field. They investigate the core tasks, applications and other important steps in constructing a successful NLG system. The field is generally split into two parts or traditions. The Text-to-text (T2T) generation method, where a new coherent text is generated from existing text. This is often used for translation between languages, summarizing text, spelling corrections and more. The other part of the field is Data-to-text (D2T) generation. That is, data is used as an input into various NLG algorithms which output a generated text that is built on the data. The data can be of a different format, such as: textual, numerical, structured or unstructured. Gatt and Krahmer also mention that text generation from visual input, *Vision-to-text*, such as image or video is a newer branch of NLG that has been less researched but is very promising, it has for example been used to translate videos to text using a factor graph to combine visual detection with language statistics[53].

Some systems are data-driven while others are not, that is a decision that is taken independently of architecture choice. That an NLG system is data-driven refers to the fact that some parts of the determination or structuring of data in a produced text are automated by using the data itself and various machine learning approaches. This is typically the case when very much data is available to train on. It can be very useful to use such techniques when rules can be found implicitly from data, or if you only want to choose the data that has certain features. Data-driven approaches are useful in various settings, an example that we will get to know better in the literature review is from [55], where tweets from traffic data are automatically generated.

For our report generation system, we will design a Data-to-Text type of NLG system. We will build the system using dynamic templates, which is chosen since we want to be able to convey data in a consistent way, with minimal requirements for explicit grammar setup. Our approach will be partly data-driven since more data will sometimes result in longer reports.

2.2. NLG Architecture

There is a variety of possible approaches to set up a fully functioning NLG system. The oldest and most common is the modular pipeline. The other variations all perform the same foundation tasks, but just in a different manner or with a different focus. In this section, we will get to know the different categories that have been identified in the literature and see what defines each of them.

2.2.1. General Modular Architecture

The typical pipeline of NLG system was originally formed by Reiter and Dale [41, 42]. Which has been used as a building block for many NLG systems which will be further discussed in this section [28, 31, 39, 50]. The pipeline consists of three major modules that each hold different tasks of a content and structural related nature. The three modules are:

i **Document planner (sometimes called Text- or Macro planner)** where the decision of ”what to say” is made.

ii **Sentence planner (or Micro planner)** where the decision of ”how to say it” is made.

iii **Linguistic realization**, where the sentences are generated in a correct manner.

A traditional NLG task is typically split up into sub-tasks and according to Gatt and Krahmer[16], which base their findings on [42] by Reiter and Dale, the sub-tasks are generally six. They are explained here below. In each task, we will take an example of a task that is relevant to our system that needs to be performed when processing raw log data and generate human readable text from them. The hope is that these examples will make the phases more comprehensive and the essence of the task will be better conveyed.

1. **Content determination**: Deciding which information to include in the text under construction.
   Here the task is to conclude what parts of the data are important and what messages we want to communicate. There is generally much more available data than is sensible to include in a written text, so some details and some information should be ignored while the rest will be taken to the next step of the pipeline. This phase will play a large role in our research since the approach for this phase needs to be specifically designed for our domain, the data and the goal we want to achieve. The goal is to inform experts on anomalies and why they happened. This will be discussed further in Section 2.5.3.

   Example: We do not want to include all requests that are processed within the system. We only want to include those that show or are related to anomalous behavior.

2. **Text structuring**: Determining in which order information will be presented in the text.
   Here the task is to conclude the order that the data should be presented in the generated text. That is, should the text be ordered temporally, by importance, by some defined graph structure or something completely different.

   Example: In anomalous reports we start with an overview of everything the report entails, then describe requests in a temporal order where all details are also listed in temporal order.

3. **Sentence aggregation**: Deciding what information to present in individual sentences.
   What data can be combined in one sentence and what not. When each data component is placed in a sentence on its own the text might become odd and unlike human-written text. There are no general rules for this phase since it highly depends on data and the domain of interest. Conditional rules need to be formed, either depending on human-defined preferences or they can be learned from gold standard representation using machine learning.

   Example: If the same type of request is performed three times, It is unnecessary to list each in one sentence, they can be combined in one sentence.

4. **Lexicalization**: Finding the right words and phrases to express information.
   Now the content has been decided and the current task is to find a way to best represent this content in natural linguistic structure. All content that has been chosen must be expressed in some way. The focus of this phase depends heavily on how random the output of the system should be. If a news article is being generated you would not want the same phrases to be used multiple times in different articles, since that is unattractive to the reader. However, if complex technical data is being interpreted it might be better to keep it as simple and consistent as possible by using static phrase choices. The same data can however both have negative and positive effects and that stand needs to be conveyed through lexicalization by complementing the words correctly.

   Example: The same behavior of the system can be explained in different ways, for example we can either say “an anomaly was detected” or “unexpected behavior occurred” to indicate that the system identified an anomaly. Either we want to use a consistent representation or vary it between instances.

5. **Referring expression generation**: Selecting the words and phrases to identify domain objects.
   This task is very closely related to the one described above. The task concerns domain entities or objects. That is, ”a discrimination task where the system needs to communicate sufficient information to distinguish one domain entity from other domain entities” [42]. Here we e.g. need to find out what properties define each entity. One of the biggest challenges of this task is when to use pronouns since it depends a lot on whether the entity of interest is in focus in the text or serves as an environmental factor and whether it has been mentioned before.

Example: We do not want to name an request by its type and command ID each time we refer to it. It is better use pronouns, such as 'it' or 'the request' if the request in question is still in focus of the narrative.

6. **Linguistic realization**: Combining all words and phrases into well-formed sentences.
   Here the actual sentences are generated, that is relevant words and phrases have been decided and now need to be combined in well-formed, naturally looking sentences. Different approaches have been proposed to do this. The most used approaches in state-of-the-art work are; human-crafted templates, human-crafted grammar-based systems and statistical approaches.

   Templates are considered most applicable when domains are relatively small, and variations should be minimal. That is the case for our research. When this approach is chosen, it is usually set up as a sentence structure with variables placed within a sentence and those variables can be filled in with the relevant words each case.

   Example:
   Template: Request *requestName* with ID: *requestID* was sent at *requestTimestamp*.
   Realized text: Request SetRoute with ID: 1234 was sent at 12:34 17.06.2020.

Typically items 1-3 fall under the Document planning module, items 4-5 under the Sentence planning module and item 6 under the Linguistic realization module.

All of these aforementioned tasks have been researched in many different settings and can be performed in various manners. The phases are not always exactly split up in the manner that has been mentioned. Depending on what is needed in each application, some are split up into more detailed/coarse phases. The main core is always present, that is the content is somehow determined and formed and presented in a natural language format. In a recent survey [45], the pipeline they have extracted from recent literature consists of a discourse planner, text schemata, rhetorical relations, surface realizer and systematic grammar. Those phases contain the same overall steps as are mentioned in the earlier pipeline design.

Depending on the system and application domains different phases get the most emphasis. Recently, most research papers focus on a subset of the pipeline.

The modular pipeline architecture is widely used due to its simplicity but there are some disadvantages to this design of task flow. According to [37], the main disadvantages are that the information flow is only one-directional through the system so there is no revision in the design flow and there is not enough communication between components. This can result in text of poor quality. Furthermore, a challenge discussed in [16] is the generation gap, which refers to decisions taken in the early parts of the workflow that can have unexpected consequences later on.

Additionally there are many more approaches possible, where the modules are not used to split up the tasks. They will not be described here, as they will not be used.

The ideology of the modular pipeline along with its core components will be used as building blocks for the NLG pipeline designed for our system. It will however be enhanced with extensions that will be further discussed in the coming sections.

2.2.2. Data-to-text Modular architecture

In 2007 Ehud Reiter presented a new architecture specifically for data-to-text systems [40]. He built on the foundation of the modular architecture he had previously introduced in 1997 with Dale [41] which was discussed in Section 2.2.1. In this architecture, he added two data processing modules, Signal Analysis and Data Interpretation. They are then followed by the same Document planning and Microplanning & Realization modules as before, see Figure 2.1. The two added modules are required since data-to-text generation systems require a large amount of data processing, that is not accounted for sufficiently in the original modules. The design presented by Reiter assumes numerical input data.

Next, we will get to know what all modules entail in this design and what functionality they are responsible for.

i **Signal Analysis:** According to Reiter, numerical and other input data is analyzed here, with the goal of finding patterns and trends. Here, noise should be excluded from the data i.e. extracting relevant data from the raw input data. He states that structured input data such as log files or

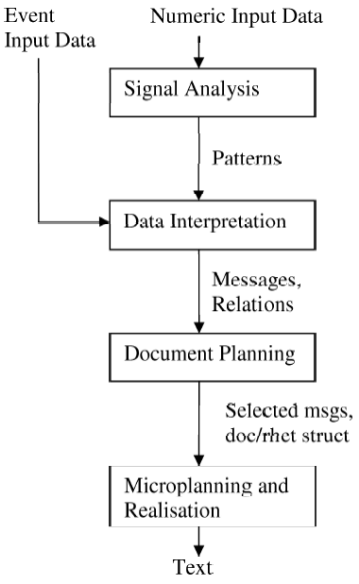


Figure 2.1: Reiter's Data-to-text architecture

discrete events can most often bypass this module. An example from Reiter is a sudden rise in some numerical values, like the rise of heart rate.

ii **Data Interpretation:** This module was included to take care of identifying more complex and domain-specific messages from the patterns and trends from the earlier module. Then creating messages from the detected patterns and assigning a level of importance to them. Identifying relations between data chunks and messages is also an important task of the module. An example of this can be to link the rise of heart rate to some other factor, e.g increase of medicine portion.

iii **Document Planning:** Here the task is to decide what events should be conveyed in the text, like in the general modular architecture. Showing too much information can cause the most important information to become unclear.

iv **Microplanning and realization:** Now, the actual expression of language is chosen to represent the previously decided content. Reiter mentions that it is viable to perform these tasks by using simple templates if that is appropriate for the users. This module can also be used to convey any uncertainty of the presented data.

This pipeline design will be a large influence on the design of our NLG pipeline as we will be designing a data-to-text NLG system. This pipeline will be enhanced with more components that will be explained better in the following Section (2.2.3).

2.2.3. NLG with other modalities

NLG is moving towards mixed modality. Modality is a term that is often used to describe the format of a medium of some sort, examples of modalities are text, speech, video and images. Different combinations of modalities are often seen in daily life, e.g. on websites, in advertisements and even sometimes in natural language applications. The research field of combining and comparing different representation modalities for data summaries and explanations has been of interest in multiple domains. Sripada [50] researched different combinations of modalities for reports created from scuba diving data. They introduced a pipeline for the creation of text and graphs from raw diving data. They showed that users considered reports using two modalities more useful to access safety conditions than the ones that only used one modality (text-only and graph-only). Additionally, they state that using annotations referring to patterns in input data helps users interpret the text.

The domain of neonatal intensive care has been a research domain of interest for data extraction and visualization. Since the domain is of course delicate and needs to have fast, precise responses for the relevant users, it is important to find out which different representation methodologies can help the

users in the best and most efficient way. Law et al. [29] compare textual and visualisation data summaries in this domain. They found that users chose more appropriate reactions given the data when presented with expert-written textual summaries than computer-generated graphics. That is, textual summaries provided a better understanding of the data. Nevertheless, graphs where subjectively preferred overall by the participants of the study. A significant difference was not detected in speed and content of responses to questions posed from the data when comparing reactions to textual and graphical data summaries. In a later study in the same domain, Portet et al. [39] presented an architecture that brings together signal processing, medical reasoning, knowledge engineering and NLG. They showed that computer-generated texts and visualizations yield comparative effectiveness. Human-generated expert texts gave better results than computer-generated ones, but despite that, they managed to show that autonomic generated textual summaries of complex, temporal data can be effective.

More recently a study was conducted to compare different information presentations for uncertain data and measure the effect on human-decision making[17]. There Gkatzia, Lemon and Rieser showed that NLG enhances decision-making under uncertainty compared to state-of-the-art representational methods. Textual reports created with NLG gave up to 24% better decision making than purely graphical ones and a combination of both resulted in a 44% increase.

Latif and Beck [28] add the use of NLG to augment map visualizations from bivariate numerical data. Their approach finds important data parts in a data analysis process and then generates text and visualizations from that data part. They use a template-based text generation approach that is based on a minimal number of parameters to tailor the properties needed in the reports. They state that the approach generalizes better to other contextual settings than many other approaches for multimodal representations. They do not include user studies to evaluate their results and declare it to be a largely open research question of which information is better represented in which modality. The same authors previously used similar techniques in another domain, for scientific publications [27]. There they found that integrated visualizations augment textual descriptions and that word-sized inline graphics are helpful to link visualization better to text. Similar research was conducted in [49] where an NLG system has been incorporated into a visualization system, their main advantage is an interactive data fact feature that helps users interpret visualizations.

Architecture similar to typical modular NLG architecture has been implemented for some multimodal systems. The one that is most related to this project is from 2014 by Mahamood et al. [31]. There, annotated graphs are generated alongside text data summaries. This is done for an NLG engine called Arria. The graph generation is integrated into the NLG pipeline and carried out in conjunction with the textual generation process. The pipeline can be found in Figure 2.2. There, the data analysis and data interpretation modules analyze the input data and produce a set of messages that can be used in the reports, the data is assumed to be numerical for this pipeline. The document planner decides which of these messages should be communicated in the report, what type of graph should be used and what data channels are displayed and annotated. The textual part is the same as in the data-to-text pipeline introduced in Figure 2.1. The visualization part is interesting, the visualization planner generates the graphs and positions the annotations. The annotation microplanner and realizer generate the actual annotation texts, and the rendering component adds the texts to graphs. Finally, the generated texts and graphs are combined in one report. They discuss that cognitive psychology is an important part to look to when generating data summaries since preferences depend on the user's cognitive intelligence. The author believes that annotated graphs can help users to identify key events more easily. The annotated graphs have shown positive results from the users of the Arria engine.

This report generation task is very similar to the task that we wish to accomplish in our research. This is because we intend to extend the NLG pipeline with graphical coponents and annotating them. The architecture structure presented in Figure 2.2 will be used as a base for the report generation pipeline for our TCS incident reports.

A summarized take-away message from the current literature on this matter is that different modalities work differently in different circumstances and the modality that a user likes the most is not necessarily the one that gives the best understanding. Most often a combination of modalities is preferred over a single one. Therefore, it might be useful to include both textual and graphical explanations in as many cases as possible for our project. These findings support the addition of graphical components to the generated anomaly reports. Furthermore, we will look into how the graphical components are received by the domain experts by comparing report versions containing graphical components to pure textual ones.

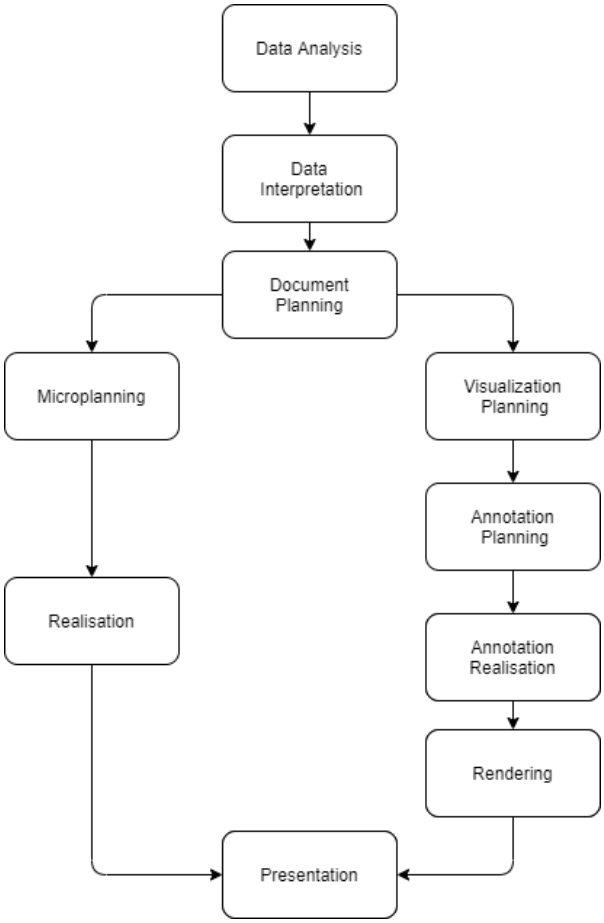


Figure 2.2: Pipeline architecture of the Arria NLG Engine

2.3. Natural Language Generation applications

As has been mentioned before, NLG systems have been applied to different levels of complexity. In [45] the versions of the existing applications are split into six categories:

1. Database content display,
2. expert system explanation,
3. speech generation,
4. limited report and letter writing,
5. automated document production and
6. presentation of information to people in an understandable fashion.

This list might not fully represent all the possibilities for application but gives a little insight into the broad possible application spectrum.

NLG has been used in multiple sports-related systems, typically to generate sports reports from statistical data [16]. Different types of sports applications exist as well, Yu et al. [64] design a video-captioning module that can narrate the details on a sports video with natural language and Tagawa and Shimada [51] propose a method to create a thorough sports summary from tweets using NLG. Plachouras et al. [38] generate financial reports, where complex data and numbers are translated into natural language reports, both on expert level and novice. NLG has also proven to be very useful in the health domain, assisting people on various levels of expertise. For example, it has been used for the generation of surgical procedures using an adaptive NLG scheme [63]. NLG has also been used to generate patient reports like is done by Jordan et al. [24], where brain imaging technology

is translated to both patient and clinical reports. NLG methodology has also been used to analyze complex technical data. Molina et al. [34] look into how the work of a human data analyst can be made easier to explain the meaning of results to end-users by generating explanatory descriptions about the meaning of quantitative sensor data, which is in no way understandable as raw data, other than to experts in some extent

Our system is meant for experts, which is sometimes the case for NLG systems. Expert systems in the general sense most frequently refer to computer systems that emulate human experts and help with decision making in complex domains. When we talk about our system being an expert system is not to be understood in this relation, but rather that it is meant to help and be used by system experts. It will indeed assist with some decision making, but more importantly with the interpretation of complex data which will then help experts make system based decisions.

NLG systems that create explanations for experts is not a new research field. According to [45] the properties that define such a system are e.g. that the system needs to do some reasoning, dynamically explain system behavior and know system rules. When the system has those things set down it also needs to take care of text and sentence planning and lexical choices.

2.3.1. Data-to-text

The Data-to-text (D2T) methodology is the most common type of NLG system and the version of interest for our project. D2T has been applied in various domains, mainly to generate different types of reports that efficiently make data readable and understandable to the reader. The domains that have been researched in depth according to Gatt and Krahmer [16] are for example sport reports [60], virtual newspapers [34], weather- [3], environmental-, financial- and health-reports [29, 39].

The data of interest in our project has both temporal and geographical factors that are important to account for in content selection and report suitably in our reports. Therefore we will look more into how data with similar features have been processed and reported using a D2T methodology.

Geo-referenced data is of interest in the RoadSafe project by Turner et al. [57]. There it is discussed that the data is challenging data since it requires linking events to their underlying geography and that mapping is not straightforward. They implement a system to explain road ice forecasts. The main challenge they find is that accounting for a large number of objects is not practical in terms of generating referring expressions for the objects. Their content selection is based on how experts perform the task in focus for the generated explanations. They manage to generate texts that are overall well received by experts but need improvements in domain reasoning and aggregation techniques.

Some systems generate reports for data that has both temporal and spatial factors that cause challenges, this is called spatio-temporal NLG. An example of this is researched by Tintarev et al. [54] where the task is to figure out how to best form feedback for volunteers trying to catch unwanted mink in the countryside of Scotland. The authors incorporate abstractions in mink sighting data across vast geographical spaces, temporal trends (time of year or day) as well as managing to encourage volunteers to continue their volunteer work. A collection of domain-specific conservation newsletters are used as guidelines to study the structure of language and lexical choices. This domain is not related to ours, but the paper points out challenges such as correctly defining reasoning over spatio-temporal facts, which is relevant to this thesis project.

Since this is the NLG category of interest for our project, we will go a little bit deeper in properties and recent advances in data-to-text generation to know what needs to be kept in mind when developing such systems. Data-to-text applications are generally split into two categories; rule-based and trainable [61]. The trainable category uses data-driven approaches, while the rule-based systems rely on manually written templates and rules for text generation. Most data-to-text make use of rule-based template filling methodology and have been shown to produce high-quality output which is generally very accurate as it is highly constrained by humans. To be able to use trainable approaches we need to have access to enough of input-output pairs for a machine learning system to be able to infer rules from the training data. Enough training data is frequently not available and that is often the reason why NLG systems are not autonomous.

van der Lee et al. [61] explore the combination of template and trainable approaches by giving statistical and deep learning-based systems templatized input and generates templatized output. They do this to attempt to come up with a good approach to tackle complex situations, which require a very extensive set of rules, which would be time-intensive to form and maintain. They wanted to see if time and money can be saved while reporting similar results as more established approaches. It showed

promising results for a noisy, heterogeneous language corpus from football games and reports.

The D2T system designed in our project will be rule-based since there is not any ground truth data available in the domain for report creation, so we do not have data to use for training. The applications given here above indicate the possibilities that D2T has to offer. We see that complex data can be handled efficiently and the incorporation of geo-referenced and temporal data is possible. We will design a system that aims to account for all those features.

2.3.2. Infrastructure for transport systems

There is not any research to be found for NLG specifically in the railway and logistics domain apart from previous research on the same TCS by Urumovska [58]. To the best of the author's knowledge the only research utilizing NLG in the railway domain is [33] where the symbolic translation is observed, from the Italian language to Italian sign language in the rail station information domain. This is not helpful to our research but worth to mention.

Nevertheless, some methodology from other NLG applications in related fields is useful, since it can provide insights into possible methodology for the railway domain. In [55] Tran and Popowich look at how NLG can be used to automatically generate messages about traffic incidents. Their approach is data-driven that works from a corpus containing examples of tweets formed in natural language. They focus on a generation system that can be applied to different contexts like road incidents. They make use of a combination of alignment model, generation model and location-based user model to be able to provide automatic location-based info to end-users.

Complex data has been used in NLG systems and the applicability of some different domains has been tested. In [32] a system is presented that uses statistical NLG techniques; it is useful for non-textual data that is stored in a tabular dataset. Their approach is built on a knowledge-light methodology, which means that it mainly uses statistical methods to generate output text.

There are some papers, like [48] that focus on how NLP can be used to extract information from incident reports, where railway incidents are often embodied. So that is in a way an inverse to our research. Vossen et. al [62] present a method to obtain a large volume of text-corpora with event data for studying identity and reference relation. That is, they derive data from reports like the previously mentioned paper. Here they work partly from a large corpus of railway incidents in Ireland. In [47] we have another example of a similar paper, but they focus on entity recognition from written incident reports. Those are examples of applications from the fields of 'Natural Language Understanding' and 'Natural Language Interpretation'. They are very related to NLG and are in a way the inverse of natural language generation since their goal is to derive meaning from a text, while the goal of NLG is to create text that holds some meaning.

The Ontologica project [9] addresses the centralized traffic control logics of the railway system of Italy. They use the notion of NLP to translate user queries to machine-readable queries. The goal of the project is to be able to represent a railway system using an ontology, to help expert users find information faster when needed and adjust or create new rules that the railway structure must adhere to. The goal of the system was to bridge a gap between a machine-readable representation of railway data and a user-friendly representation of the same data. Which sounds very close to the goal of our project of interest. They choose to build an ontology rather than a relational database because they needed a more expressive formalism than they could get in a database. Ontologies are also good for enhancing keyword searches. An extension of the system, to create automatically human-readable descriptions of the ontology structure was mentioned as a possible future direction of the project but, I have not found any traces of that yet.

We have seen the possibility for successfully modeling the railway infrastructure for natural language-related processing. It is however not possible to apply these methodologies straight to our project since their applications have other properties and focus on different aspects than we will be focusing on.

2.4. Case-Based reasoning

Case Based Reasoning (CBR), sometimes called Case-Based learning is a problem solving and learning methodology where the idea is that data batches (cases) often have similar characteristics and can, therefore, be treated similarly with regards to the goal of interest. We believe that similar anomalous cases should likely be handled similarly in our case. Therefore this methodology caught the attention of the author. Instead of looking into typical classification algorithms for the task of finding a suitable

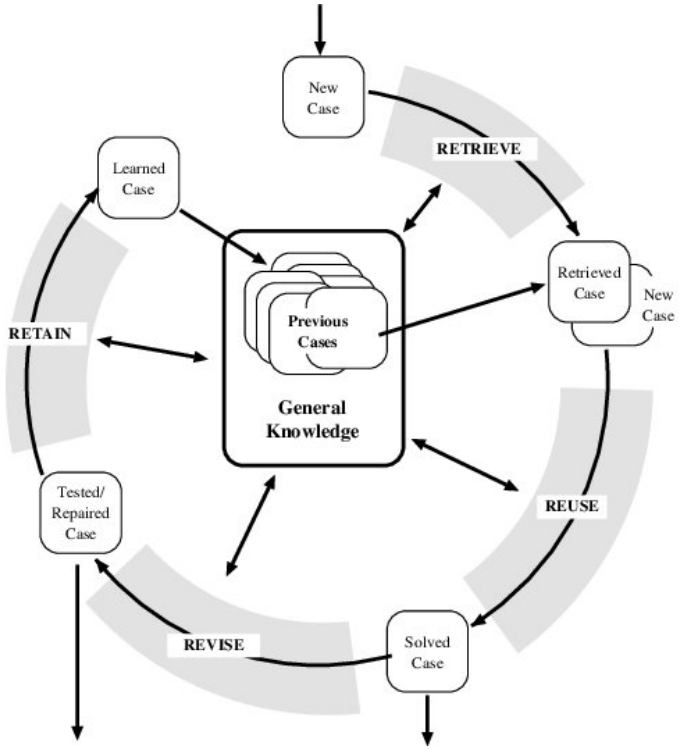


Figure 2.3: The CBR according to Aamodt and Plaza [2]

report presentation for each case, we wanted to incorporate a technique that continuously learns and accounts for revision of solution as part of its design. CBR is a sub-field of machine learning. There, knowledge of previously handled cases is utilized and applied to new cases that need to be solved. This leads to an incremental learning tool since the system gains more knowledge by every new case it handles. CBR was first introduced in 1983 by Schank [46], but in, another paper by Aamodt and Plaza [2], a good overview of the methodology is presented along with a process model that has been widely used ever since. There, the authors describe CBR in one sentence: ”To solve a new problem by remembering a previous situation and by reusing information and knowledge of that situation”. By incorporating this, a system tries to approach the human mindset by using similar responses for similar data input, in the same way that we humans often present the same concept with a similar composition of words and a doctor treats similar medical symptoms with similar procedures.

The task process cycle, called the CBR cycle, explained in [2] can be seen in Figure 2.3. The four main processes are said to be the four Rs:

1. **Retrieve** the most similar case or cases from a case base.

   - sub-tasks: Identify, search, initial match, select.

2. **Reuse** the information and knowledge in the retrieved case to solve the new problem. It needs to be decided which parts can be reused and what needs to be adjusted.

   - sub-tasks: Copy, adapt.

3. **Revise** the proposed solution by applying the solution to a problem and evaluating the fitness of the revised solution by a domain metric or expert evaluation. If the evaluation achieves sufficient results, the solution is retained.

   - sub-tasks: Evaluate solution, repair fault

4. **Retain** the parts of this experience likely to be useful for future solving, the case base has a new, solved case to learn from.

   - sub-tasks: Integrate, Index and extract

A disadvantage of this methodology is that during the building of a case base, each revision needs to be evaluated and revised. It can be a challenging part of CBR systems to come up with methods for all tasks to meet the requirements of the domain of interest. Before going through the process cycle, the current problem situation must be identified and encoded in the same way as the stored cases are encoded. This is so some similarity measure can be applied to find relevant cases from the case base. The general knowledge is domain-dependent and can be in the form of rules or some sort of domain model.

CBR has been used as part of NLG pipelines with good results. The use of CBR typically has the advantage that there is not a need for an exhaustive set of templates, since the methodology operates by searching for the most similar case and adjusting it if needed.

CBR is utilised for template selection for knowledge-intensive NLG template selection in [21]. The templates are for the lexicalization and surface realization of the report generation process. The goal of the implemented system is to 'Achieve coverage of a broad range of messages by combining instances of a restricted set of templates, providing automated means for dealing with overlaps between the information conveyed by the templates found and ensure coherent use of context information.' This is a goal they manage to fulfill. A case retrieval net is used as a memory model and a concept taxonomy used to find similarity between any two words. They help to make the search for the most similar case efficient since they work on approximation. The domain of interest in the research is storytelling, so they calculate similarity by storyline properties and the semantic proximity of terms according to common ancestors.

The methodology is again used for NLG purposes in [3] where wind weather forecasts are the domain of interest. The incentive to use CBR for this task is that similar weather conditions should have similar forecasts. They claim their methodology is knowledge-light since they do not need much expert information about the system to implement the forecasts, but they can only do that since there is a lot of data available in weather data and the relevant forecast. They use the Java Library jCOLIBRI[1] which is a CBR framework that provides generic modules for all the stages of CBR. That proved to be helpful for their implementation. In this domain a case consists of weather attribute values and times, therefore similarity of cases is calculated by wind strength, directions and timing of the day. The requirements for a match in the retrieval component are considerably strict, which turned out to be a small problem for the system. Therefore, they propose a fuzzy matching accompanied by more revision rules. The revision component uses expert rules, learned during post-edit tasks by experts, to ensure writing conventions and rules of the domain. When new adjustments have been reviewed that case is added to the case base, which helps the system become more comprehensive. The evaluation of the system is solely implemented with automatic metrics. The system has other trainable NLG systems that outperform the CBR system in terms of NIST scores, which is an automated evaluation metric of generated text. Since it outperforms an NLG system that had previously gotten the best evaluation from human experts, the authors hypothesize that their system would also do well with human evaluators.

CBR has not yet been researched in the rail domain. The different incident cases in the rail domain can be defined by a set of properties, similar to the weather report cases in [3]. Thus we believe that our domain, like the weather domain, is suitable for applying the CBR methodology. The methodology from [3] will be adjusted to our domain and influence the design of a CBR module for the pipeline of the system. We will incorporate the relevant case properties. This CBR design will be further discussed in Section 5.7

2.5. Processing raw data contained in log files

The rail industry is both a producer and user of data [56]. That is, a very large volume of data is produced during a normal day of train management. That data can be used to study and learn from to gain more insight into the system. The sizes of the produced data files are enormous, so it is not possible to do so efficiently by hand. Therefore, suitable methodologies need to be found and applied to help with this study and analysis of the generated data. This is a process we need to perform for the content determination phase of our NLG pipeline.

Log files are generated automatically by the system, they hold information about all events that occur within the system; requests, responses, status updates and much more. In this subsection, we will investigate different ways to make sense of unstructured data from log files. To be able to detect anomalies on our data we need to start by being able to process the data from our log files and represent

them in some sort of data structure. In the following section we get familiar with data modeling and processing methodologies and identify the ones that are most suitable for us to apply to our data.

2.5.1. Log mining

To be able to learn patterns and graphs from data we need to perform some data mining on the available log files. *Data mining*, is a process where structural patterns are identified in data, often containing links and paths between data points. Log mining is a sub-field of data mining where this process is applied to log files. There are different ways to process the data contained in log files and detect different interesting aspects of the data. Some methods are automatic, while other methods are manual or a blend of both.

Some papers have been published where the mining of railway data or other similar and related data structures is done [18, 19, 25]. Of those papers, [18, 25] are most applicable to our project since they use process mining to perform an analysis on train path conflicts from log data generated in the Dutch train describer system, TROTS. Describer system is another name for a system that keeps track of train positions using their identification numbers, signaling messages and interlocking systems. Their project is therefore very related to the content determination phase of our NLG pipeline and can provide valuable insights for us. Even though a direct methodology can not be applied since the designs of the systems of interest are not compatible. A state diagram that explained the conflict loop is used to identify the conflict, which worked out well for them. They explain their results to experts using only visual output, while we will instead be using natural language enhanced with visual components.

Anomaly detection for a complex system related to ours is carried out by Hadžiosmanović et al. [19]. The goal of their anomaly detection is to detect cyber threats where illegitimate actions are performed. Their approach is semi-autonomous. It demonstrates data mining on SCADA data which is an acronym for “supervisory control and data acquisition”, which is a group of systems that train control systems are considered to be a part of. They mine the data of the water management system of the Netherlands with regards to anomalous log traces. They focus on detecting legitimate user activities that harm the system. They emphasize the importance of extensive knowledge on the application domain, for log mining to be possible. They perform system analysis by themselves and with process engineers familiar with the system to gain that knowledge. The methodology used in the paper is building a task analysis tree that holds all possible user actions. The authors of [19] built their analysis tool called MELISSA, which manages to extract non-frequent patterns, which they assume to be suspicious activity. That is, they employ pattern-based analysis, which is often called pattern mining. A limitation of their work is that they only focus on identifying single anomalous actions, not sequences or paths of actions.

2.5.2. Data representation and rule formulation in related domains

The domain of rail infrastructure does have more research interest than one would initially assume, there are even whole conferences dedicated to the field i.e. RSSRail[1]. Large parts of the research are on planning the train schedules, which is not in the interest of our research. There are some useful insights on data representation and modeling, that can be found in this literature.

Ontology In a review paper by Turner et al. [56], a relatively recent increase in the usage of ontologies and semantic technology is discussed, which can be useful to draw meaning from data. A successful example of that can be found in [9]. There, an ontology is built to be able to represent rules for the railway system of Italy. Since trains are set to move by certain tracks, they are very susceptible to rules, which are also crucial for the safe movement of the trains. Therefore, it is possible to specify explicit logical rules that all train movements need to adhere to. Since that is the case, it should also be possible to set up logical rules of commands within the train control system itself. By doing that we would be identifying the only allowed ways for commands to travel through the system and all other transitions would be considered to be a fault in the system.

State Machines Recent research has shown that the usage of formal methods and tools, such as state machines, is considered favorable for modeling railway control systems [14, 15]. Formal methods are mostly used for formal specification and verification of the systems.

[1] https://conferences.ncl.ac.uk/rssrail2019/

In [15] the results from a questionnaire answered by the participants of the RSSRail conference 2017 [2] show that usage of state machines to represent state relationships in railway software, is used a considerable amount with favorable results. State machines are a type of network or graph that represents the possible flow through the states of the system.

In the previous research on our project, Urumovska [58] worked with state machines to represent the regular paths of commands and their response through the system. There it was suggested that graph representation should preferably be learned automatically, rather than modeled by hand.

Model building Some research has been done concerning data representation in related domains to ours. Technologies to recognize patterns from data in an automatic way, e.g. by the means of data clustering and signal processing methods are relevant to this project. An example of that can be found in the GraRep project [10], where the model learns low dimensional vectors to combine them in a graph. They show promising results and have their tool available for external use. The main disadvantage of their approach is the large computational cost of the modeling.

Han et al. [20] analyses a Chinese train control system for failures. They do that by modeling the scenario leading up to the noticed failure to understand what has happened. That event model is built from log files generated within the system, to attempt to reproduce the failure. The log files give insight to all steps and events that lead to failure. Failure analysis is carried out using model-checking techniques, where temporal properties are important since they allow for a flow through the system to be constructed. The authors manage to create a good anomaly explanation of their test case, which is the usage of emergency breaks on a certain train. Some disadvantages can be detected in their system approach. Records included in the created model need to be manually selected. The precision of the resulting failure reproduction of the system depends a lot on how many records are chosen and how large of a temporal interval is chosen. They can also only find failures once they have happened, not prevent them altogether.

Model-checking is again used by Ledru et al. [30], now for modeling and verification of signaling rules in the train control design. The signaling rules are deployed in a signaling system, which tries to make sure that accidents can not occur on its premises when cooperation between physical devices, policies and agents are executed. The model they propose contains the main constraints needed for the flow of a TCS, but not as precisely as is needed for the setting of our project.

The methodologies mentioned in this subsection are meant for modeling a whole system and being able to verify security measures. That is not the plan of our current research, since we are focusing on being able to detect specified incidents in a TCS and explaining the behavior leading to the incidents. Despite showing favorable results when explaining anomalies found in one log file type in [58] using state machines, that is not a feasible data structure for our project. This is because the data relationships are too complex and can not be modeled efficiently and completely by a state machine. We will, therefore, build a model that fits the system design and domain rules of our TCS, so that all system components and data relationships that are in the scope of this thesis project are included. That will be satisfactory for our setting, while also being simpler to implement and make use of.

2.5.3. Detecting Anomalies

There are different kinds of detection strategies to find cases where data behavior is not as expected, that is, anomalous cases. An anomaly is a deviation from the common rule, type, arrangement or form of data. Jakub and Branišová [23] present a method for anomaly detection in log files. It is based on dynamic rule creation for data mining techniques. There, three types of detection techniques are listed. *Anomaly detection*, which first creates a system profile of proper functionality, then it looks through gathered data to check if non-standard behaviors are present and pinpoints them. *Misuse detection*, is a sub-technique of anomaly detection where various system profiles of “wrong” functionality are set up and a match to them is found in data and marked as an anomaly. This has proven to be a successful way to detect known anomalous scenarios in the sub-parts in the relevant log entries. Lastly, *Hybrid detection*, which is a combination of the other two strategies, where both safe and unsafe profiles are created and applied in the profile matches in data are sought out.

Anomaly detection is the best choice when all possible anomalies can not be foreseen. When new anomalous scenarios can occur and when we do not have all examples of correct scenarios. In our

[2] https://dblp.org/db/conf/rssrail/rssrail2017

case, the system processes are too complex to be able to model all correct ones, so anomaly detection is not chosen as the detection strategy for the system. *Misuse detection* is a very useful tool if some finite set of anomalies is defined and needs to be detected. We will make use of *misuse detection* to identify anomalous scenarios in our work and its application will be further discussed and explained in Section 5.3.2.

Some challenges are common in anomaly detection strategies and need to be kept in mind. According to Hadžiosmanović et al. [19], anomaly-based approaches generally all have similar limitations. These limitations are connected to the model that is composed of the expected behavior of the system. That is, the model that is used to present the normal scenarios is not perfect. That can happen when some scenarios are left out by accident in the process of modeling correct behavior. Therefore, some anomalies are not caught or allowed behavior can be marked as an anomaly. Both of these defects can result in safety concerns in the anomaly detection system. These challenges are applicable for our modeling of system behaviors since we will be working with modeling anomalous processes and detecting occurrences matching them.

Anomalies within a railway system can be caused by various reasons, such as code bugs, flawed designs, railway parts malfunctioning or miscommunication between different components of the system. When a system malfunctions in any way it is vital that system experts can understand why, so that can be fixed. Which makes sure that the malfunction will not occur again for the same reasons. This is a challenging task as is mentioned in [20]. There, the reasons mentioned are that is is hard to construct a model for a complex system and it is challenging to draw useful information from a mass of raw log files. This task is often referred to as *anomaly description* in the general sense and is an important part of anomaly detection but is often overlooked in research projects. It covers interpretation and sense-making of a detected anomaly. The main challenges are how to dig up important information regarding the anomaly from the dataset. Another challenge that is mentioned by Akoglu et al. [4], is the decision of the time frame that is relevant to the anomaly and how connected data points need to be to the position of the anomaly to be relevant to its interpretation. Those challenges are present in our TCS as well and we will be accounting for them in our design.

2.6. Evaluation methods of automatically generated text

Efficiently defining evaluation procedures for NLG systems is still a challenge in the field. Evaluation of NLG systems is an essential part of the NLG pipeline. It helps us understand the quality of the system and makes it possible to compare systems. To be able to compare any two systems in a good manner they need to be evaluated using a similar methodology. This has been somewhat of a problem within the field of NLG since there is no consensus on how these systems should be evaluated. The main reason for that is related to the input of NLG systems. The input can be very diverse and there are multiple correct outputs possible for each input set. That is e.g. because there are very many ways to phrase the same information.

It can be very challenging to efficiently evaluate NLG systems since language can be very subjective and exposed to high variability [6]. Human evaluations and automatic metrical evaluation are the two main categories used for evaluating automatically created text. According to Gatt and Krahmer [16] the correlation between metric and human judgments appears to differ a bit across studies, suggesting variation between domains, algorithms and datasets. Different parts of NLG systems can be evaluated separately and using different methodologies. REG algorithms are often evaluated using automatic metrics, whereas content determination might need to be evaluated by experts that understand what is expected of the system. Gatt and Krahmer [16] draw the main conclusions regarding the evaluation that the choice of method has a direct impact on how the results should be interpreted. Moreover, Amidei et al. [7] discuss the need for researchers to be thorough when reporting evaluation studies, including examples, and sharing all evaluation guidelines and data.

2.6.1. Automatic metrical evaluation

The use of automatic metrical evaluation is widely used for the evaluation of NLG systems since it is relatively simple and quick to carry out. It is much less expensive than human evaluation and can easily be repeated if needed. In later years it has been debated whether this application field (NLG) and evaluation technique make a good pair. This is for example discussed in [59] where two main reasons for their controversial usability are summarized. The main reason they pose is that automatic metrics

are uninterpretable, meaning that a low score according to a metric does not help pinpoint what the exact problem is in a generated text, so an error analysis needs to be performed to solve that. Secondly, automatic metrics do not correlate with human evaluations, so they are not a good match to evaluate linguistic properties, since they are meant to reflect human linguistics. Even though the usage of the automatic metrics is controversial, van der Lee et al. [59] still report that 80% of NLG papers published in the two biggest NLG conferences reported evaluation results using automatic metrics.

The most used metrics are ROUGE, METEOR and BLUE. Novikova et al. [35] discuss the need for other metrics than the typical automatic metrics like BLEU[36]. The argument they pose is that for a metric to be sensible to use for NLG evaluation, it must be sufficiently correlated with human preferences, which they point out is rarely the case. They perform a detailed error analysis that shows that the automatic metrics have a hard time distinguishing between medium and good quality. They also conclude in saying that the metrics can still be useful in detecting poor performance, so they can be used to some extent with good results.

Even though they can theoretically be used, it is not as easy as it sounds for NLG systems. To be able to use automatic metrical evaluation methodologies, there needs to be a gold standard available for the task that you are evaluating, that is the golden, correct output for a given set of data. In the setting of our project, there is no gold standard available so we will not be able to use this evaluation method. Indeed, according to Amidei et al. [6], no gold standard is available for most cases of NLG applications. This is the case for our project since no data has been previously handled that can serve as a gold standard.

2.6.2. Human evaluation

Human evaluation is a common method to evaluate automatically generated text. Human evaluation can take many forms, recently the current best practices were summarized by van der Lee et al. [59]. They state that for a general assessment of NLG system quality, human evaluation has been and remains the gold standard. Evaluation can for instance be carried out through an evaluation survey or interviews, and questions can be either fixed or open. Different implementations call for different evaluation and processing of the evaluation results.

There are many aspects in human evaluation that can influence the outcome of evaluation greatly and therefore need to be reported and included in the design of the evaluation procedure. The demographics of the pool should be reported and of course the number of participants. Furthermore, the inter-annotator agreement (IAA) should be reported, since it holds valuable information that can not be directly derived from the normally reported results. It can also be minimized by training judges and giving them guidelines on how to judge text and systems [16]. This can be costly when the pool is large and diverse. Amidei et al. [6] summarize the use of IAA over the last 10 years. In this paper, attention is drawn to the main drawbacks and challenges that can arise in human annotation and evaluation. The stability, accuracy and reproducibility of the annotators are aspects that give insightful information on how reliable their outcome is. IAA is a vital part of estimating these aspects and as is pointed out by Amidei et al. [6], it is not used as much as it should. The main challenges with regards to IAA to look out for are: sufficiently explaining the experiments of collecting the data, naming the coefficient that is being used and choosing one that is suitable for the data collection. The setup of the annotation procedure is often not explained well enough in research papers, that is whether the annotators were working alone or not in their evaluation task. Amidei et al. [6] furthermore suggests that correlation coefficients should be used alongside kappa statistics to get more insights into how reliable the evaluation really is.

Human evaluation is the most applicable evaluation methodology for our project. We will therefore study more details about its application and what challenges need to be considered.

Criteria and questions

There is no single correct way to evaluate NLG systems. In 2018, van der Lee et al. [59] created a summary of evaluation criteria used in the papers published in the two biggest NLG conferences; INLG[3] and ACL[4]. The most frequently used criteria notions were fluency, naturalness and quality, but there where many other diverse notions as well that were considered as relevant for the evaluation purposes. The main concern is that the criteria used needs to be well defined, so even if there are diverse words chosen to represent the criteria, the reader needs to know what criteria are being measured. We will

[3] https://www.aclweb.org/anthology/venues/inlg/
[4] https://www.aclweb.org/anthology/venues/acl/

measure the quality of the system but not fluency and naturalness explicitly. We will rather focus on the readability and understandability of the reports.

Human evaluation can be split into intrinsic and extrinsic evaluation [16]. According to van der Lee et al. [59], extrinsic evaluation is noticeably uncommonly used, even though it is considered more useful. Let us look into what is inherent in the difference between those two categories:

- **Intrinsic**: Measures the performance of the system without relation to the actual goal of the system. It focuses on the properties of the output of the system. The evaluation criteria can i.e be human likeness, clarity, fluency, readability, relevance, correctness, compatibility of text amongst others.

- **Extrinsic**: Focuses on the impact or fitness for the specified purpose of the system. The most advantageous but most time- and cost-intensive evaluation method. This methodology is most applicable when a whole NLG pipeline is being generated and the adequacy of the evaluator pool is certain. One example of an extrinsic measure is the time needed to read or understand the text. Another example could be how well reports manage to provide understanding.

The main goal for the system being designed in this thesis is to explain anomalous incidents in a TCS. We want the output reports to be informative and readable. This is a goal defined by the main CGI supervisor of the project. This is not an uncommon goal description for an NLG report to aim to fulfill. Hommes et al. [22] have a similar goal, where reports are generated to inform cancer patients about their medical data effectively and clearly. There, the main evaluation criteria are the relevance of the report to the domain and data, they want interpretation to be easy and reports should be consistent between cases. We can see that even though the domain of that system is different from ours, it still has similar report properties it aims to fulfill.

The questions posed for human evaluators can be for comparing items or judging them individually by the selected criteria. It is important to design the questions well with regard to the insights you want to gain from them and what is easier and clearer to the respondents of the questions. Open questions can be useful when systems are still in an implementation process and can be improved. But closed questions, that have a set of possible answers are good for evaluating the final result of systems.

Pool of evaluators

There is a difference in techniques that should be used depending on whether the evaluation is expert-driven, or reader-driven. When the human evaluation is reader-based, it is often carried out by the usage of crowdsourcing, since that gives the possibility of a diverse and big pool. In the case of our research, we will be creating the texts for experts, so the experts and readers are the same pool of people. Typically, more general readers are required than experts to be able to make a statistically valid assumption. Expert users are experts in the domain of interest. They have a good skill-set and knowledge in relation to the domain. In the case of this research, the domain is the train control system ASTRIS. However, the experts of interest for this research are not expert end-users of the system, but experts that work on making and maintaining the system.

Evaluation scales

The choice of scales to elicit judgment can impact the results. It needs to be chosen carefully and taken into account when results are concluded. According to Gatt and Krahmer [16], discrete, ordinal scales are the dominant choice. However, continuous scales should be taken into consideration since they can give more distinctive judgment. Amidei et al. [7] discuss the use of different scales in NLG evaluation tasks. They specifically point out what is typically done wrong in the application of these scales and how they can be used more suitably. Likert and rating scales are two different versions of evaluation scales that are often confused. Both scales can be used to estimate feelings, opinions and attitudes of responders towards a specific matter. Amidei et al. [7] review the difference between rating scales and Likert scales. Likert scales are aggregate scales that typically offer the choice of points that vary from strongly agreeing to something to strongly disagreeing to the same thing. They are meant to collectively capture the phenomenon under analysis by getting ratings on multiple aspects, Rating scales, on the other hand, are meant to be used on single items and can be graphical, numerical or comparative, Rating scales are often called Likert scales in papers and that should be avoided since they should be treated differently. Since Likert scales are designed to capture collective results on a certain item,

their results from the items should be summed or averaged to produce a total score for each item. According to [59] 7-point scales are found to maximize the reliability, validity and discrimination in answers. Despite this 5-point scales are the more popular.

When making use of Likert scales, the difference between offered answers can be perceived differently by respondents. To make sense of that it is good to use some centrality measures to measure the central tendency of the answers.

Scales are often analyzed without justification for their statistical choices. Since there is no one right way to interpret these scales, all choices made must be explained and justified, both for explainability and reproducibility. According to Amidei et al. [7], the main reasons for doing this is that using parametric statistics for ordinal data can result in skewed results that are not valid. It can reduce statistical power and the difference of answers can be either under- or overestimated.

In this project, we choose to use 7-point rating scales for all rating questions. The 7-point format was chosen to be able to maximize reliability, validity and discrimination in which the 7-point scales are found to do well. The rating scales were chosen rather than Likert scales since we do not consider it suitable in our case to format questions on a disagree/agree scale. Instead, a relevant rating scale for each question is chosen.

2.7. Conclusion

In this chapter, we studied the main tasks that are needed to perform this research and implement a successful NLG system for experts in a complex domain. We have especially explored the different tasks that compose typical NLG pipelines and discussed how it is possible to fit raw log files from a train control system in such a pipeline. This can be done by processing the files, modeling the data and extracting content that is useful to convey in a report using anomaly detection. We have seen that the incorporation of multimodality can be favorable for improving the performance of automatically generated reports. Furthermore, we have become familiar with the CBR methodology and how that can be utilized for the task of automatically generating reports. Finally, we discussed how best to evaluate an NLG system like ours.

2.8. Research gap

We have identified a gap in existing research that we will attempt to fill by performing this research.

We investigate generating explanatory reports using complex data that needs to be extensively processed and has temporal and geographical features that need to be accounted for. To our knowledge, this is among the most complex data to be processed with an NLG methodology. The data requires extensive log analysis and domain rule incorporation to be able to process and match data correctly.

Previous literature has not considered selecting different modalities for generated reports based on the content being explained. This is a gap we aim to address in our work. That will be done by providing new research insights by looking into the possibility of utilizing CBR to optimize the report generation by choosing presentation templates. Furthermore, we will investigate how user properties, such as their expertise, influence their preference for modality presentations and if that should be accounted for when choosing presentation templates for the reports.

All of this will be done in a domain that has not been researched much in the NLG perspective. However, the domain is important and safety-critical. We will be using data from a complex TCS to automatically create detailed reports for domain experts. We have a focus on log analysis and data interpretation in this project. Even though some NLG methods have been applied with transport systems, none have looked into the effects of different visualization components on the preference and understandability of the reports.

3

Setting of the project

The domain of this thesis project is specialized and needs some introductions to provide context and understanding of the needs and requirements of the system being researched. In this chapter, we will become familiar with ASTRIS, a Train Control System (TCS) layer, which produces the data processed in the report generation system of this thesis. We aim to help analyze the data by automating and visualizing anomalous scenarios of the system. The domain and the problem of interest were provided by GGI NEDERLAND B.V. (CGI). The problem is a current real-world problem.

All facts and knowledge about the system explained in this chapter is originated from the system specification documents[8, 44] or other sources within CGI, e.g. project descriptions and informal interviews with supervisors.

In Section 3.1 we will learn what a TCS is and what its purpose is. Section 3.2 will explain the architecture of ASTRIS to be able to understand the extent of this software layer, followed by Section 3.3 where the data produced by ASTRIS is introduced and the content explained. We will conclude this chapter with a discussion of the problem we intend to address in this thesis in Section 3.5.

3.1. Train Control System

ProRail is an organization that takes care of all infrastructure that is related to the trains in the Netherlands. That entails stations, railways and everything besides the trains themselves and the people working on them. Train dispatchers are employees of ProRail and operate the railways in the Netherlands. They organize the train traffic and make sure everything goes according to plan. They are responsible for the cost-effective movement of trains as well as other on-track railroad equipment and to optimize the movements and plans of the trains. The train dispatchers have to make sure all elements of the railways are set correctly so the trains can run on railways without any trouble. This has to happen according to a process plan containing rules about specific times for initializing train routes. Most of this planning and scheduling happens in an automated way, but when some trouble occurs, the train dispatchers need to be able to react fast to make sure it will not build up to a bigger problem, like delays or other defects. All commands that the train controller gives are sent to ASTRIS, which is a program maintained by CGI. If a problem is found or an incident occurs, this will be analyzed by experts of ProRail and CGI, using the ASTRIS log files to find a possible cause. Those experts will be the users of the system designed in this thesis.

ASTRIS is a train controlling software layer written by CGI. The control layer is business- and safety-critical as its goal is to make sure that movements of trains are safe from one station to another and more precisely from one railway section to the next. Train dispatchers can issue commands with confidence and can rely on the railway data that is displayed through software on their screens. When ASTRIS receives a command, it checks whether the current state of the railway allows for the new command to be carried out. ASTRIS checks the current state of the infrastructure and that the incoming command will be accepted by the underlying interlocking system, that is, it will not cause a problem for any operation that is in operation. Furthermore, it checks if switches are in the correct position according to requirements, traffic lights are in good condition and that there is no blocking condition or other obstructions. That is, it checks whether all safety requirements are met. If the command passes all

safety checks, it is allowed to continue through the TCS. ASTRIS then sends the command to the next layer of the TCS of the Netherlands, which is the interlocking system, that performs the actual operations. For instance, the next layer can then change positions of switches or turn train signals green. All of this requires high precision and proficiency since a TCS is a safety-critical application that handles life dependent infrastructure and ensures all requirements that are essential for the safe movement of the trains are met.

ASTRIS communicates information back to the end-users of the TCS to ensure that they always have correct information about the state of the railway network. The information communicated back entails i.e. whether a railway section is occupied or not and what the current state of traffic lights and position of switches is.

ASTRIS is composed of many different software components that each have an important role in the command structure for operating and managing the railways. Each of the components generates a very precise log file that holds all information about performed events, such as requests to the component and responses to those requests. Most requests that are sent to ASTRIS are on a route level, that is, they apply to a defined part of the railway that holds multiple rail infrastructure elements that all need to work together to ensure that the railway part (route) is in the correct condition to be used by trains. Depending on the type of underlying interlocking system and the command given, When ASTRIS receives a request, it translates it to different commands on an element level. The element level commands are in terms of actual switches, signals or other element types. When the route or element level requests are finally performed, ASTRIS communicates information regarding the state of the railway tracks back to the train controller so their state is known and up-to-date.

Elements are rail infrastructure objects. The term “Element” covers twelve element types. They all have different purposes and behave differently. They are therefore represented differently in the log files as well. The most frequently mentioned are switches, signals and sections. To make sure that the reader knows what those elements are, Figure 3.1 has been included. Switches and signals are more generally known terms, while sections are a defined sub-parts of the railway.

(a) Switch (b) Signal (c) Section is the area between two white blocks

Figure 3.1: Most frequently mentioned element types.

3.2. Software Description

In this section, we will cover the main functionalities of the core components of ASTRIS. According to ASTRIS’ system specification document [44], the main reason for the component structure of the software is to be able to clearly distinguish between operating and signaling functionality within the system. The functionality of all the main system components of interest for this research will be described. We will learn which of them produce the log files that are included in our system model and how they connect. We will also introduce important information contained in configuration files that describe connections of railway routes and elements. These connections are vital for the linking of data between software components.

3.2.1. Software Architecture

I **Process management layer** This is the system layer that the train controller operates. This program sends requests to ASTRIS, which performs the next steps.

II **Routing component - ARC** This component is the heart of ASTRIS, it receives all commands that are sent to ASTRIS and determines what component should receive it to process it further.

This component communicates with all the other core components in ASTRIS.

Log files of type 1 in section 3.3.1 contain logs from this component. They are one of the log file types that are of interest for the report creation system designed and created in this research project

III **Route component** This component handles the requests that concern railway routes. Those are the most common requests of the system and most often regard preparing a route for a train to pass it, revoke such a command or blocking a certain element from being used.

When a route request is set into motion, a chain of sub-commands is initiated. That is, requests are sent to other components to get a route correctly set. The route component starts by sending requests to the element component to get the current states of specific elements. If the elements are not in the desired state the route component sends commands to change the states as needed.

- **Information distribution component for routes - IDCR** Handles state updates for the routes that are processed in the route component. This component generates log files that are of interest for the report creation system of this thesis. Log files of type 4 in Section 3.3.1 contain logs from this component.

IV **Element component** This component handles all commands that regard a single element. Elements can be various parts of the railway, e.g. switches, traffic signs, railway sections or bridges. It most often receives the sub-command of the route component. Sometimes commands are given directly to a single element by the train dispatchers and then this component gets the command straight from the routering component.

- **Information distribution component for elements - IDCE** Handles state updates for the individual elements that are processed in the element component. This component generates log files that are of interest for the report creation system of this thesis. Log files of type 2 in Section 3.3.1 contain logs from this component.

V **System & area components** The system component handles instructions for unusual elements that do not fall into the classic element class. It has a connected component, 'Information distribution component for system' (IDCS) that handles its state updates. The area component handles restrictions to specific areas or sections of the railway. It has a connected component, 'Information distribution component for areas' (IDCG) that handles its state updates.

These two components are not directly in focus of our research, they are however important for the overall system architecture and are therefore just mentioned here.

VI **Management component - Meldingen** When ASTRIS is started up it is the task of this managing component to start all the other components up and make sure everything is in order. Additionally, it continuously watches over them and checks if all the components are still running. Whenever one of the other components runs into an error, this component generates the error message. Log files of type 3 in Section 3.3.1 contains logs from this component.

VII **Safety Component - BevNL** This layer does not contain much business logic. Its function is to transfer commands out of ASTRIS to an underlying safety software, that sends the commands to the actual physical infrastructure of the railway, like switches and lights. There the commands will be carried through. Log files of type 3 in Section 3.3.1 contains logs from this component.

VIII **Underlying safety layer** This layer is an underlying security system, also referred to as interlocking system, that ASTRIS communicates with when it assumes that commands are safe to conduct. It is a collective name of all the security implementations that communicate with ASTRIS. This layer can in some cases reject an instruction that is sent to it, then a response containing the reason will be returned to ASTRIS. There are multiple versions of the underlying security layer. Thus communication with this layer is not always structured the same way. some safety layers can receive route requests, while others can only handle element requests. Then ASTRIS needs to translate all route requests on an element level.

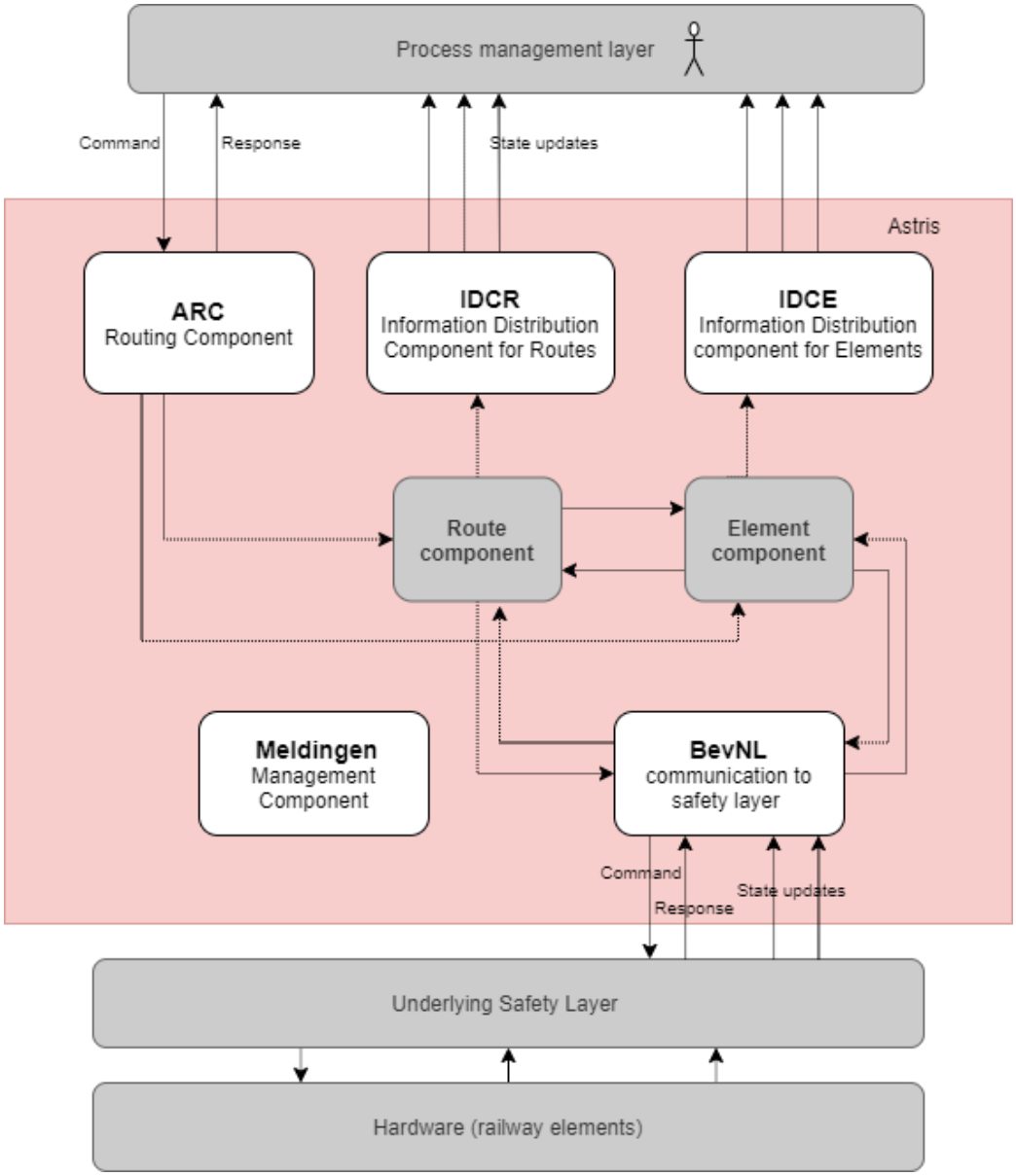


Figure 3.2: Simplified Component Diagram of ASTRIS and its connected layers. The red box represents ASTRIS and the white components are in focus in this thesis project

The components are continuously communicating together. To understand how the communication is structured we explain the typical structure of a single command and its aftermath. The structure of the instructions is always as can be seen below, as long as everything is working correctly.

1. *Request* is sent. Example: “Change the position of switch x to position y”.
2. *Response*. tells you if it is possible or not to carry out this command. Example: “It is possible to change the position of switch x to the left position”.
3. *State updates*. Updates on the actual state when the command is being applied. Can be multiple state updates to one command. Example: “The switch is now in an intermediate state’ and then ’the switch is now in position y”.

This is a series of events we see constantly repeated in multiple places in ASTRIS. In Figure 3.2 we see this behavior illustrated twice, first between the process management layer and ASTRIS and secondly between the BevNL component and the underlying safety layer

This series of events can happen multiple times for each request, depending on the complexity of the request. Requests can have multiple stages or sub-requests, where each stage performs a specific task that needs to be finished before the next one can be carried out. Each sub-request has a trace in the log files along with its state updates. To give an example of the complexity of a request we look into the main stages of a request to set a route. First all, elements need to be reserved or allocated for use. Next, elements such as switches and level crossings are placed in their correct position. Next, the light signals turn green to indicate that the train is allowed to move across the railway of the route. Finally, the train moves and the railway sections turn occupied one by one.

3.2.2. Configuration files

In addition to the actual software components there are **configuration files** used by the system that maintain the connection between railway elements and other setups. ASTRIS makes use of several configuration files in its daily operations. Two of them are used to ensure the correct behavior of our report generation system and the main information used from them is described here below.

- **Route configuration file** These configuration files contain information about all routes in the railway network. The most important information to mention is regarding what elements are related to each route. Each route includes and makes use of multiple elements and each element may be a part of multiple routes. That is, the relationship between routes and elements is many-to-many.

  A route consists of a set of elements. Those are considered the elements *in* the route, i.e. railway sections, traffic signals and switches. Each route is identified by its starting signal, end signal and its LR-String, which is a sequence of the required positions for its switches. Additionally, some elements are *related to* the route. They are not a part of the railway that the train needs to directly use to finish its movement and pass over the railway part, but the element nevertheless has a requirement to be in a specific position for everything to work correctly.

- **Element configuration file** These configuration files concern setup for all elements. They specify which routes all elements are related to, and what the requirements for each element are for each route. Elements are identified with three main properties, their type, name and the area they are positioned in.

  Elements can additionally have specific relationships to other elements which are all listed in the configuration files. Two elements can be coupled together, making so-called *coupled elements*. This means that if one element changes position, the other element is required to do so as well. This most often applies to switches, since they can block off a part of the railway if nearby switches are positioned in opposite directions. A different property an element can have is to be part of a *Niet profiel vrij situatie*, henceforth called NPV, that means that the element is in a part of the railway where a situation can occur where the necessary space around a train to be able to assure safe movement overlaps with the space of another element.

To explain the configuration properties of elements and routes we will use an example from the system. For security reasons, we will not use the correct names of elements, but all facts are taken directly from the setup found in the system. This example setup can be seen in Figure 3.3. Railway sections are marked with two small white blocks crossing the railway, switches are on crossroads and the traffic signals are marked with the **O-|** sign.

The green and yellow lines represent two different railway routes that both have signal 0, sections 1T and 2T and switch 1A in common. For one route the switch needs to be in its right position and for the other route is required to be in its left position. Each of the routes contains numerous elements. To illustrate further, the green route in Figure 3.3 contains sections 0T,1T,2T,3T,4T and 5T, starts with signal 0, ends with signal 10 and uses switches 1A and 7.

Switches 1A and 1B are coupled, which means that their behavior is dependent on each other. If one changes position the other also needs to change to ensure that they are always compatible with each other.

Switches 1A and 3 are both part of an NPV situation with regards to section 7T. That means that for section 7T to be considered safe, it requires both of the aforementioned switches to be in a specific position. Those positions ensure that no other train can come within the movement profile of the train passing over section 7T.

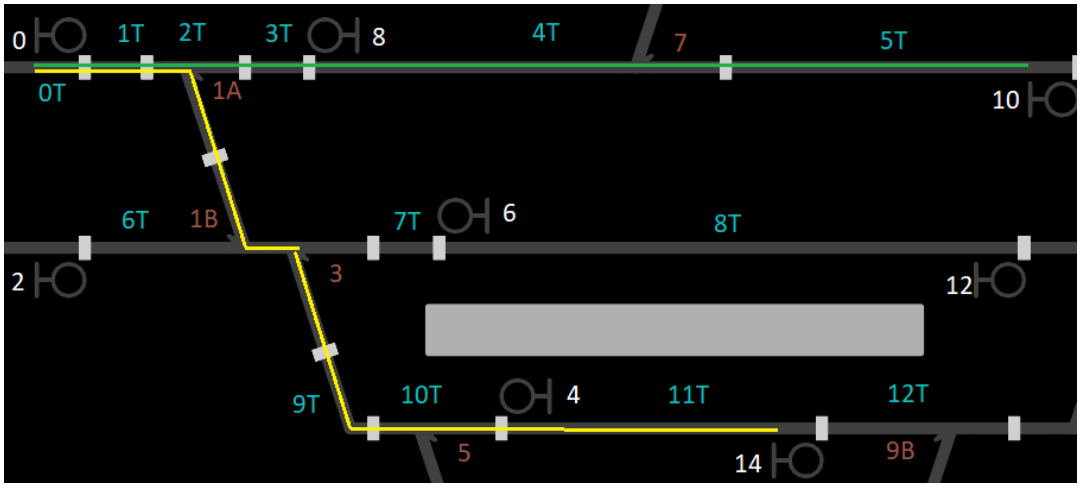


Figure 3.3: Example of railway overview. The gray lines represent the railway tracks, switches are marked with a brown color, sections with cyan and signals with white.

To give an example of the many purposes for each element we will look more closely into switch 1A. It is part of 11 different routes. Six of which require the switch to be in the position right, and five that require it to be in the position left. Furthermore, it is a related element for one route, where it is not part of the actual railway that the train uses, but still needs to be in a specific position for the train to pass safely.

3.3. Data produced by the system

All events, such as errors, requests, responses and state updates performed by or in relation to ASTRIS are stored as log file entries. They hold very detailed information that describes all requested and performed changes. Most event logs used in this study are split into a header and a body. The header of the entries contains, amongst other data, information on what action is being performed, entry identification, the sender, receiver and timestamp of the entry. The body of the entries contains information that is specific to the action being processed. Such as route and element identification and the very detailed information on the current state of the entity in question in multiple key-value pairs. The information contained in the body is differently structured for each action being performed in each log file type.

Most log files that are generated by the system are stored in SOAP XML envelopes. Each file contains multiple log entries which moreover have multiple keys and values. One log file type we will focus, Meldingen, deviates from these rules, there the log entries are represented with a long line of information found in the log files. They do not have a key-value structure and need different processing. That will be discussed further in Section 5.2 where the data processing is described. All data found in the log files is in Dutch.

3.3.1. Types of log files

Five categories of log files are most important for the system. Therefore, we will be including them as the raw data analyzed in our report generation system. We will refer to the log files as the name of the system components that generate them as depicted in Figure 3.2.

1. **ARC** log files hold all updates concerning the requests and responses within the routing component. That shows the communication straight from the train controller to other components of ASTRIS. This file contains very vital core information on what requests are sent into the system and should be tracked through the system.

2. **Meldingen** or error log files generated from the management component. These log files are important as they contain all the error messages caused by any of the other components.

3. **BevNL** log files hold log entries that cover all information that travels from the safety component of ASTRIS and the underlying security layer. These log files contain status updates, requests

and responses going in both directions between ASTRIS and the underlying safety layer. There are different notations used for data entries depending on the versions of the safety layers.

4. **IDCR (Routes information distribution)** This log file type holds the most important information, which are events processed on a route level. It was analyzed in the research in a previous master thesis concerning ASTRIS by Urumovska [58].

5. **IDCE (Elements information distribution)** This log file type contains event data about all requests, responses and state updates for elements. The log files are much larger and more detailed than the IDCR files. Incorporating IDCE, in combination with IDCR, provides a deeper understanding of processes and anomalies within the system.

Four information distribution components are mentioned in the software architecture section (3.2.1). They are IDCR, IDCE, IDCG and IDCS. IDCG and IDCS are not extensive. Thus it is deemed unnecessary to incorporate them into the report generation system designed in this thesis. Moreover, IDCR and IDCE contain the most vital information of the information distribution log files.

The data found in these log files can be used to follow most requests through the system and find information about their responses, sub-commands and state updates. The data keys that can be found in log entries are diverse depending on which of the five log file types is in question. Furthermore, the keys vary substantially depending on other aspects. For example:

- whether the log entry contains data about a response, request, or a status update,
- what type of element is being handled,
- what the status of the element/route is.

It is difficult to make a connection between entries of different log types together as there are no one-to-one relationships that can be used to match all entries correctly together. For the data mapping, it is most important to map on the element or route identification and the time stamp. That requires caution since there can be more than one request being processed for the same element or route at the same time. The timestamp of entries is an important indicator of connections as well as command identifications. The command identifications can time out, that is they stop being included after a certain time and therefore need to be handled cautiously.

3.3.2. Extent of log files

The Netherlands is split into different areas with regard to its railway network. There are twelve areas and each has a control station that runs ASTRIS. Each control station has approximately five instances of ASTRIS running. That sums up to roughly 60 instances and each of them has its set of log files generated each day. The amount of data that is generated from each is about 50 MB. That size differs between days and instances. This sums up to a large amount of data that needs to be processed. To give an indication of the amount of data produced per day, each ASTRIS instance generates roughly:

- 15.000 ARC entries
- 1.000.000 IDCR entries
- 2.500.000 IDCE entries
- 500.000 BevNL entries
- 250 alerts (Meldingen) entries

There are many log files of each type from each day in every instance of ASTRIS. The files are sequential so all log files within an instance, of the same type, can in theory be concatenated in one alignment. That would result in a huge file that is not feasible for processing.

Although each log type is split into multiple files per type, the size of each log file is very big which makes them very hard to go through and work with. Despite that, it is often necessary since they give important insights into incidents that have occurred or might occur in the future. Currently, it is difficult to structure and search in the logging.

As all information about the current status of the system needs to be communicated to all system outlets operating the system, that means that the same data is sent to many different places. All these data transmissions are logged as log entries, which results in log entries being repeated, just with a different receiver each time.

3.4. Anomalous incidents in ASTRIS

The goal of our system is to explain anomalous incidents in a TCS. Anomalous incidents in this setting can be diverse and stem from different causes. The official definition of the word anomaly is: “Anomaly is a deviation from the common rule, type, arrangement or form of data” according to dictionary.com [1].

We have listed up three main anomaly versions that occur in the system according to an informal interview with the main supervisor of the project, and expert of ASTRIS. They indicate what needs to be detected in the system data and why.

- **System malfunction** which most often takes form of a malfunction or miscommunication within or between different layers of the whole TCS of the Netherlands.
- **Commands rejected** often by the safety layer which is an indication that there are some requirements not handled perfectly within ASTRIS. In a case like this, the command has already been approved by ASTRIS and sent further to the underlying safety layer. ASTRIS aims to account for all safety measures that the safety layer has, to never send requests to the safety layer that are rejected there.

  Commands can be rejected within ASTRIS as well. Which happens if the command does not meet requirement standards set by ASTRIS or an entity needed for the command fails to reach a required position.
- **Malfunction of hardware** on tracks, which can be caused by an element that is stuck in a specific position for some reason, i.e. due to a foreign object on the rail track or a switch frozen solid.

3.5. Problem description

When a software incident occurs within ASTRIS, it needs to be analyzed for the experts to understand what caused the incident. Currently, such an incident analysis is performed manually. This is a difficult, time-consuming procedure where log files are investigated and different types of log entries are matched together. As this is time-consuming it is of course costly and causes a safety-related process to take a long time. The logging is complex and is stored in many different log file types. This makes the log data hard to structure and query.

Now that a core understanding of the functionality of the TCS has been described, we can better formulate and explain the scope of the current research and the problem we aim to address. We aim to make the analysis process needed for anomalous incidents more efficient by automatically generating reports. The objective is to be able to use these reports along with the current manual analysis methods. That will help the experts to identify a starting point from which they can perform a deeper analysis if needed.

We will be working with and explaining information from the five log file types listed in the Section 3.3.1. These log file types were chosen since they cover the main functionality of the system. Furthermore, we have one type for each of the main components of the system which allows for a complete report generation that refers to all main parts of the system. We will need to perform substantial data mappings to be able to connect log entries from these five different log file types correctly together. This is a difficult task as they are all structured differently. Moreover, as mentioned in Section 3.2.1 each request can have multiple phases which further increases the amount of data needed to process. We utilize information found in the configuration files described in Section 3.2.2 to link together data entries from each log file type according to the setup of the railway. Each mapping will need to be formed differently depending on what request is being processed, using timestamps, route and element identifications along with command identification when possible.

From the set of data we have mapped together, we aim to detect anomalous incidents. The anomalies need to be uncovered by detecting various anomalous data patterns which will be further discussed

[1] https://www.dictionary.com/

in Section 5.3.2. To find out what incidents are most important to focus on, for what purpose and how to explain them, we will perform an initial requirement analysis. The analysis will be discussed in Chapter 4.

From detected anomalous incidents, we will create reports that explain the detected anomalies by detailing what lead up to the anomalous incident. We will investigate what presentation modalities provide the highest understanding for the experts of the system. One of the main challenges of the project is to find a good representation of the system to work with the data contained in the log files so that log entries can be matched correctly to provide all necessary information in connection with a specific anomalous incident. This will be further discussed in Sections 5.5 and 5.6.

4

Requirement Analysis

In this chapter, we will become acquainted with the steps taken to acquire domain insights for the needs and requirements of the report generation system. The viewpoints and opinions of the prospective users of the report generation system will be gathered. The prospective users in question are the experts of the TCS.

In an initial questionnaire described in Section 4.1, the experts provide an understanding of what the most important features are to consider when generating anomaly reports for the TCS. Various properties of the system experts are gathered and analyzed in subsection 4.1.1, to better apprehend the composition of characteristics of the pool of experts. Sections 4.1.2 and 4.2 will introduce what incidents the experts think are most important to explain. We will lastly introduce the system requirements observed from the answers from the initial questionnaire and open interviews with the main CGI supervisor of the project.

4.1. Initial Study

We have now studied the core domain knowledge and gotten to know the data that will be processed in this thesis. To understand the needs and expectations of expert users, a requirement analysis in the form of an initial study is carried out. The study is performed using an online questionnaire format. The pool of participants is composed of all the people that are working on developing, maintaining, or testing the TCS of interest. The pool consists of 19 potential evaluators, 14 of which answered the questionnaire.

The purpose of this questionnaire was twofold. Firstly, to get familiar with expertise related properties of the future users of the explanation system. Secondly, to gain more insight into what properties are most important to convey in the reports concerning the domain.

The choice of a questionnaire instead of interviews was made to make it easier for the experts to participate, as they are all busy with their work. With a questionnaire we also made sure that all participants got the same amount of info about the project. The study was approved by the Human Research Ethics Committee of TU Delft. All participants consented to participate after having read through a consent form tailored to the setting of the project. The consent form was approved by the ethics committee as well.

A pilot study was conducted with 3 participants of the 19 participant pool. From that questionnaire two problems where identified. The first problem was regarding a question meant for the researcher to see what anomalous events are most important to explain in an automatically created report. At first, this question was completely open so that bias would not be present in the answers, depending on what choice options where given in the question. However, that format was considered to be too complex by the pilot participants. Therefore a new version with 8 options in a checkbox layout was added, where there was also the option to add a new answer option if wanted. To not lose the reasoning behind the choices, a second question was added to ask the participants to explain their choices. The second problem detected in the pilot study resulted in a reformat of a question.

The initial study is split into four parts. The first part is a short introduction to the project. Secondly, the gathering of consent to participate. The next part concerned the professional user properties of the

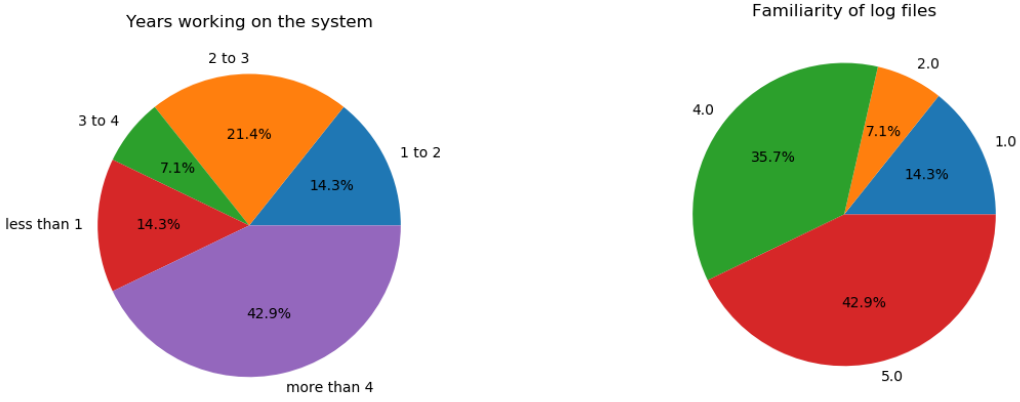
expert participants. Finally, questions regarding the preferences of the experts and their expectations of the system.

4.1.1. Expert properties

Three questions are included to be able to form a small expert model focusing on the professional properties of the experts. By doing this we gain statistics about the pool of experts and can see if there is a connection between the expert properties and their preferences for the system.

These expert questions included in the study are:

- "How long have you been working with/in relation to ASTRIS?" The options given are yearly ranges from less than one year to more than 4 years.
- "How familiar are you with the contents of ASTRIS log files?". This question was asked on a 5-point rating scale varying from "Not very familiar" to "Very familiar".
- "What is your role in relation to ASTRIS?" For this question, four main titles were provided for the experts to choose from. They could choose more than one or add roles if nothing applied to them.



(a) Work experience in years in relation to the system

(b) Familiarity of log files on a 5-point scale from *Not very familiar*(1) to *Very familiar*(5)

Figure 4.1: Distribution of expert properties

The statistics for the outcome from each question can be seen in Figures 4.1 and 4.2. There we see that 42.9% of the experts have worked on the system for 4 years or more. We also see that the majority has a good familiarity of the log files, where 35.7% have a familiarity of 4 and 42.9 % have a familiarity of 5 on a 1-5 point scale varying from "Not very familiar" to "Very familiar". The roles of the experts are various, but testers, developers and solution architects form the majority of our expert pool. Note that 17 roles are chosen despite only 14 experts answering the questionnaire. This is due to the possibility of choosing more than one role.

By looking into the connection between the answers of these three questions in Figure 4.3 we see that there are not any obvious dominant profiles. The answers to the three questions result in different combinations. The only combination we see more than once is for a Developer that has worked on the system for more than 4 years and is very familiar to the log files. Nevertheless, it is possible to detect some clusters in expert properties. They do not rely on the role of the experts, since that does not seem to have a large correlation with the other two factors. Three clusters were found using k-means clustering methodology, using k=3. K-means clustering finds clusters that minimize the within-variance of a cluster[52]. Since we have a small number of data points this simple clustering method was chosen. In Figure 4.3 the three clusters are shown and annotated with G1, G2 and G3, representing Group 1,2 and 3.

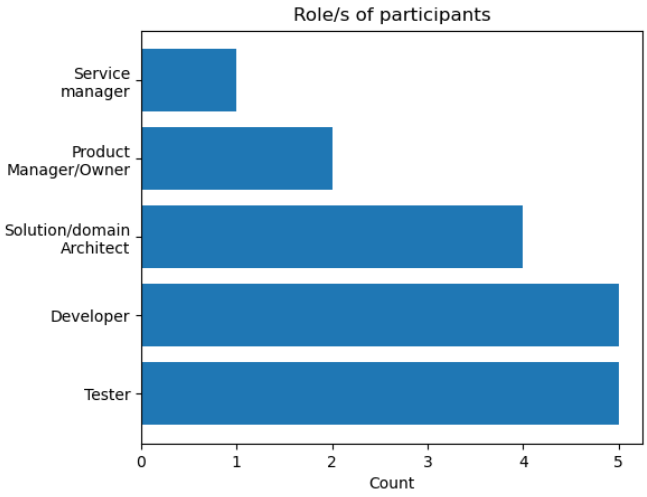


Figure 4.2: Distribution of roles of experts.

- Group 1 consists of experts that have a high familiarity with the log files and less work experience .
- Group 2 consists of experts that have both high familiarity and great work experience.
- Group 3 consists of experts that have low familiarity with the log files.

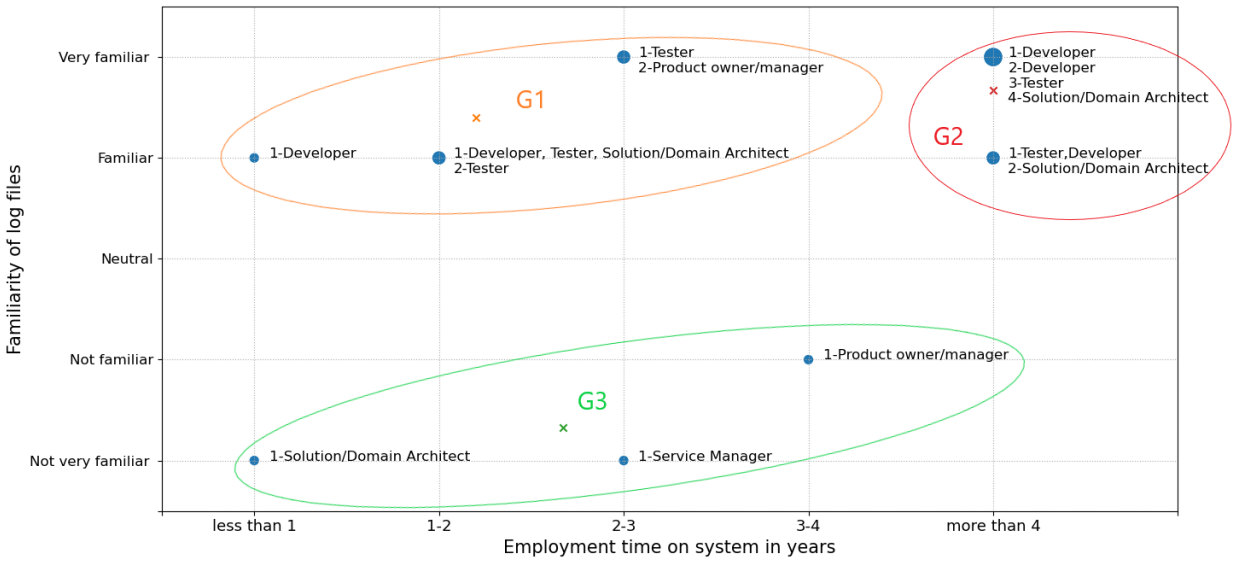


Figure 4.3: Distribution of expert user familiarity and work experience along with their roles. Three clusters are detected, each cluster's center is depicted with an *x*.

Those are the three groups of expert property combinations that are thought to be most similar. In Chapter 6, Evaluation, we will make use of these expert groups, when we look into whether there is a difference in the preferences of generated reports for experts depending on their expert properties and the groups they belong to. These expert properties and groups will help us answer research sub-question 2: *"To what extent do expert preferences of report presentations associate with their expert profiles in terms of data familiarity and work experience?"*.

4.1.2. Domain insights and expert preferences

Several questions were posed to the experts to structure the needs, preferences, and utility of automatically generated anomaly reports. The answers to those questions contain important domain insights from the majority of the actual users of our system.

Usage of reports

First of all, a question was posed to understand how the explanatory anomaly reports will be used by the experts. There the experts were asked why and when they would be likely to find a use for such a report. They could pick more than one answer and add their options as well. The results can be found in Figure 4.4. There we can see that the most common usage will be to understand defects that have happened and to learn from this previous anomalous behavior of the system and account for that in the system operations. There is no clear trend in the answers depending on what expert groups they origin from.

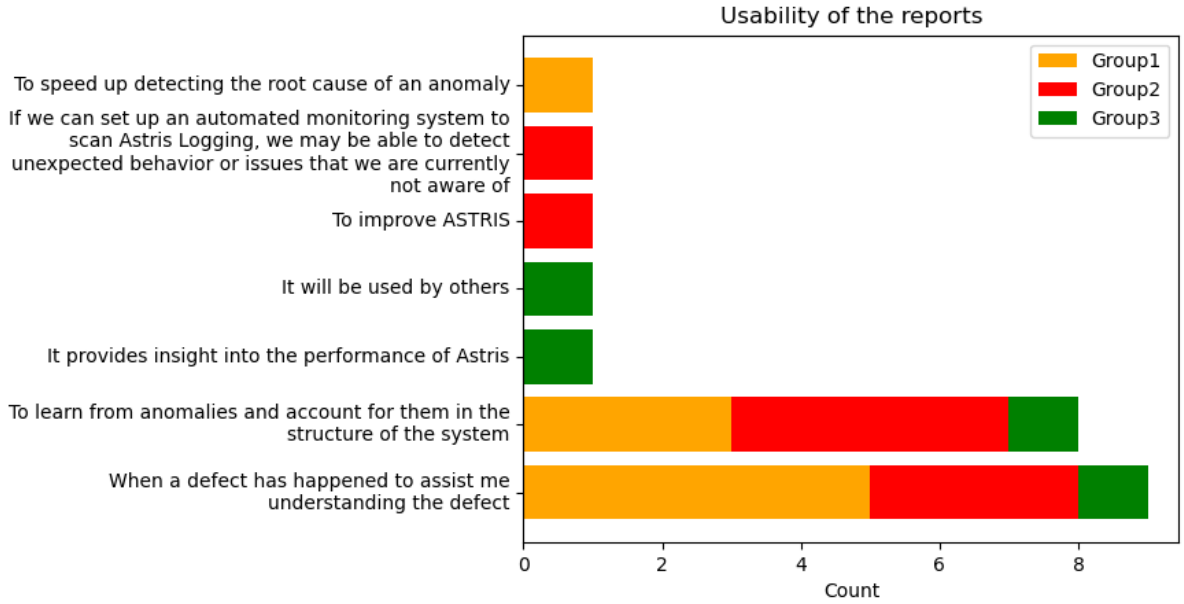


Figure 4.4: Usability of reports, grouped according to defined expert groups

Advantages of reports

An additional open question where the experts were asked what the main advantages of these automatic anomaly explanation reports could be. This question was added to understand what the experts want to yield from the system, to be able to detect requirements for the report generation. The question was kept completely open to make sure the experts were not biased toward any specific answer and thereby get insights in detail that had maybe not been in focus in the project until now.

The main trends have been extracted from the answers, where we ended up with very similar answers as were offered in the question described here above. Two trends are by far most frequent. Seven out of the 14 experts that answered mentioned faster or quicker analysis of problems that saves time as the main advantage. Six experts mention help in understanding problems and unexpected behavior. For example when the origin of the problem is originated in a different part of the system from the actual unexpected behavior.

Some advantages are mentioned less frequently. The following are a few examples. The reports can help detect anomalies that might be missed in manual detection. They make problems more approachable and directly available. With the generation of such reports less manual work is needed on the log files. In a combination with automated monitoring of created log files, it can help measure how well ASTRIS functions, by identifying bugs of other unintended behavior. By creating such reports we will have some trend analysis and prediction possibilities, as well as service-level reports. Lastly, such reports are said to be helpful to train and assist new ASTRIS employees.

All of these answers show us different aspects of when the reports will be useful for experts. Moreover, they show us what the experts expect to gain from using the reports. These aspects will be integrated into the requirements of the system.

Important events
The experts were asked to give their professional opinion on what the most important TCS incident types are. They were asked to pick the three most important events to handle that can originate from the log file types of interest for our project. Several options were given, with an additional open option, where the experts could add a new incident type if needed. The results can be found in Figure 4.5. There we see how the answers are distributed across the expert groups. Choices for all event categories are relatively evenly distributed between the expert groups.

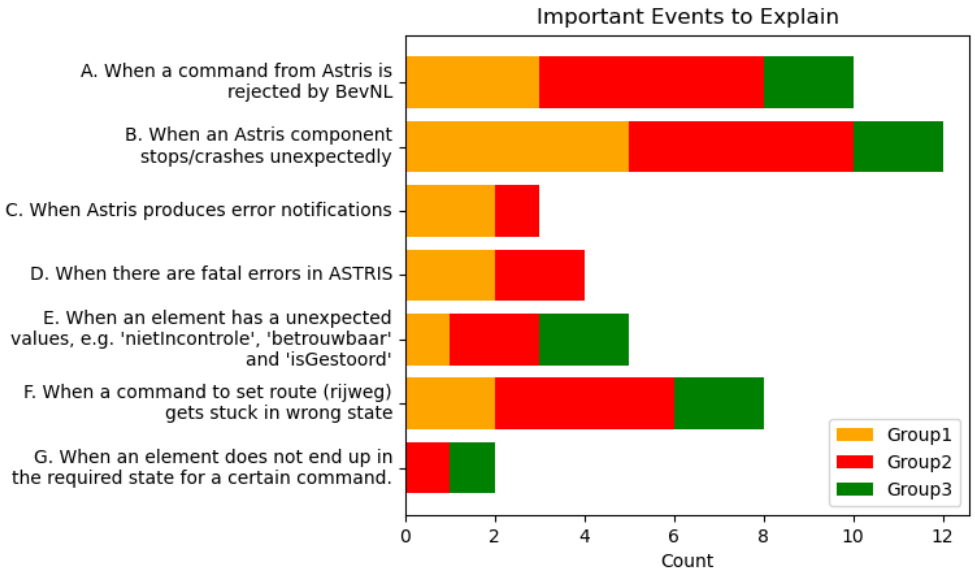


Figure 4.5: Most important events to explain, grouped according to defined expert groups

To get a better understanding and insights into why these events are important to be explained by reports, the experts were asked to justify and explain their choices. The explanations for each event category have been combined into explanatory paragraphs for each of the incident categories.

A. **When a command from ASTRIS is rejected by BevNL**

   The pre-testing of requirements should not let commands through that will not be executed. If a command has passed the requirements tests of ASTRIS but was then rejected by the underlying safety layer it suggests that the requirement setup, that is the business logic of ASTRIS, is not correct, since it should account for all the requirements of the safety layer. This incident is not visible to the end-user (train dispatchers) and can lead to loss of functionality and delayed trains.

B. **When an ASTRIS component stops/crashes unexpectedly**

   It is important to explain these crashes since ASTRIS should not stop unexpectedly as that can cause serious consequences. Log files are a good way to use as a basis for such explanations since the correlation and sequence of the messages they hold are a way to uncover the cause of these stops of components. This is useful for helping to analyze an error that has occurred. Information on why these stops occurred can be extracted from the 'meldingen' log file. An event like this indicates that there is an unforeseen combination of inputs/situations(either in business logic or code). These events give a good indication of how unconstrained ASTRIS is.

C. **When ASTRIS produces error notifications**

Errors suggest that there is a bug in the system, which should be found. Errors indicate some reasons why the infrastructure is not sound. This helps with infrastructure maintenance by contractors. Similarly to event *A*, this incident is not visible to the train controllers and can lead to loss of functionality and delayed trains.

D. **When there are fatal errors in ASTRIS**

Fatal errors are never allowed or expected, so it is good to understand why they happened. This is mostly covered by B as the result of a fatal error is that the relevant component stops/crashes.

E. **When an element has unexpected values, e.g. 'nietIncontrole', 'betrouwbaar' and 'isGestoord'**

Is important because these values are the core content of ASTRIS-messages. These values are often deduced or set long ago, errors can often be found by looking at these values. Unexpected data from the security layer leads to unexpected states in the TCS and their origin should be explained.

F. **When a command to set route gets stuck in a wrong state**

Similarly as for events A and C, this incident is not visible to the train controllers and can lead to loss of functionality and delayed trains. These events indicate possible bugs, unforeseen events, or software faults. This is important to explain because every state has a specific set of requirements to reach that state. Unexpected behavior can be determined and interpreted by the events that lead to that behavior.

G. **When an element does not end up in the required state for a certain command**

This is connected to event F, since elements are often impacted by, or the cause for a route not reaching a desired state. Errors in element states indicate anomalies and help direct their analysis in the right direction.

Some experts furthermore gave more of opinions relating the events. They are however debatable and should not be considered as the statements. One expert noted his that events A, C and D are less important than the other ones since ASTRIS produces a lot of 'expected' or 'allowed' errors. Focusing on them can clutter up the important errors. Others said that useful information on event types C and D could not be extracted from the log file types of interest. Events E and G were also said to be less important by one expert because these problems originate outside ASTRIS. It was additionally stated that event types E, F and G are the cause of B, C and D respectively. Events A, B and F are problems that are said to be often discussed by the ASTRIS team and are therefore important to explain.

By this summary, we can see that the viewpoints and opinions of the experts can be very diverse. Therefore, it will be interesting to see how their user profile corresponds to their opinions.

Language preferences

A question was posed to gain insight into the language preferences of the experts. 'Would you prefer technical values, such as state-names, error messages from the security layer and log file attributes to be in English or Dutch?'. In Figure 4.6 it can be seen that no participant chose English. This is likely due to the ASTRIS requirements and domain vocabulary being in Dutch. The split between Dutch and neutral preference was completely even, but since there was only a preference for Dutch, not English, the technical attributes in the generated reports will be in Dutch.

Modality preferences

To get an initial understanding of what presentation modality is preferred in the reports, the question "What type of presentation do you think you would prefer?" was posed. Images of example reports were presented to the experts, to make sure that the understanding of different presentations was present. The results from that question can be found in Figure 4.7a. Interestingly, no expert thought a purely graphical presentation would be better than text, but the vast majority expects that a combination of the graphical and textual modalities will give the best performance.

To find out what type of graphical presentation is informative for the experts, they were asked to choose what kind of graphical presentation they thought they would prefer. The question was set up as a multiple-choice, where an open option was also available. The answered varied greatly, the only

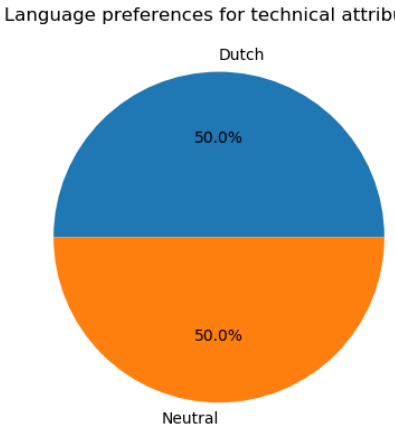
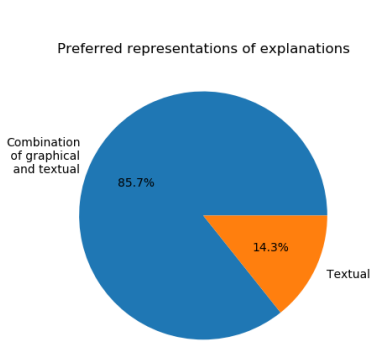
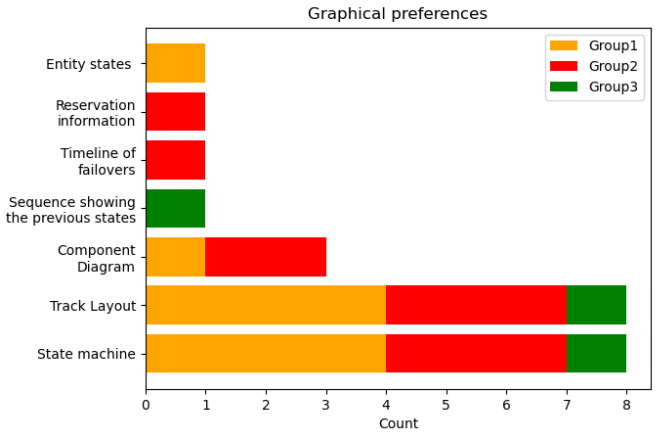


Figure 4.6: Language preferences of technical attributes

two answers that were chosen more than once are: “Combination of track layout and state machine for commands” which appeared four times, and “Combination of component diagram of ASTRIS and state machine for commands” which appeared twice. To get an overall sense of the diverse answers, they were broken down into the presentation entities they contained. Two of the 14 answers were not included since they did not contain any concrete answer to the question. These results can be found visualized in Figure 4.7b. Where we see that state machines and track layout get the highest number of overall votes. The votes for both graphical components have the same distribution across expert groups.



(a) Presentation preferences



(b) Preferences of graphical presentations, grouped according to defined expert groups

Figure 4.7: Representation preferences of evaluators

4.2. Anomalous incidents to handle in explanatory reports

From Figure 4.5 we see that according to the experts the four most important events to explain by processing the related log files are events A, B, E and F. Thus, these four events will be used to define the four following respective anomaly types.

1. When an ASTRIS component stops/crashes unexpectedly.

2. When a command From ASTRIS if rejected by BevNL.
3. When a command to set route gets stuck in the wrong state.
4. When an element has unexpected values, that leads to wrong behavior of the element.

These categories of anomalies will consequently be the focus of this project. They will be better defined by their requirements in the following section. Together, they cover the main anomalous incidents in ASTRIS introduced in Section 3.4. Anomaly type 1 is a *System Malfunction*, anomaly types 2 and 3 cover *command rejection*s. Lastly, anomaly type 4 covers the main abnormalities caused by *malfunction of the hardware*.

Other anomalous events that did not rank as high will also be included if they are related to or influence one of the main anomaly types. It is often the case that many anomalous events are linked together. While we do not explicitly define the other events as specific anomaly types, they are often a part of detected anomalous incidents.

4.3. System Requirements

The initial study provided good insights and supported determining the focus of the project. We now want to define the requirements for all incident types as functional requirements and then we will also look into the non-functional requirements of the system.

4.3.1. Functional Requirements

First of all, we need to be able to detect the anomalous incidents from the raw log files. We then need to be able to explain the behavior of the system that leads to the anomaly.

After having distinguished the anomaly case types as the most important anomaly types to handle, they were defined formally by the CGI thesis supervisor. This is done to have a clear definition for the anomalous case, so that can be used both in the report generation phase for including the relevant information and at evaluation time for evaluators to get an understanding of what the reports should be explaining.

Anomaly type 1 - *When an ASTRIS component stops/crashes unexpectedly.*

In a case like this, we want to know what was happening in the system for the last 4 minutes before the crash. We want to know about all commands executed and be informed in detail about all situations that could lead to some sort of failure. We want to be aware of detected errors that have occurred during the time frame of interest.

Anomaly type 2 - *When a command from ASTRIS if rejected by BevNL.*

We want to see all information related to the command that got denied. That is, we want to see other commands that make use of elements that are also part of the denied command in the last 4 minutes before the denial/rejection. We want to be informed in detail about all situations that could lead to some sort of failure. We want to be aware of detected errors that have occurred during the time frame of interest.

If this anomaly type is detected, it can be an indication that something is incorrectly handled in the requirements of ASTRIS, since ASTRIS runs checks before sending it to BevNL. That is done to try to confirm that everything is correct and up to the standard of the underlying safety system before it is sent there.

Anomaly type 3 - *When a command to set route gets stuck in the wrong state.*

We want to know the specifics of the command. We want to know what state the route got stuck in. We want to know about other related commands that are being executed the last 4 minutes before the rejection. We want to be informed in detail about all situations that could lead to some sort of failure. We want to be aware of detected errors that have occurred during the time frame of interest.

Anomaly type 4 - *When an element has unexpected values, that leads to wrong behavior of the element.*

We want to know all the relevant information about the element in questions. We want to know what commands the element is a part of during the four minutes before the unexpected behavior. We want to know the specifics of those commands. We want to be informed in detail about all situations that could lead to some sort of failure. We want to be aware of errors that have occurred during the time frame of interest.

For the rest of the thesis, these anomaly type names will be used to distinguish the anomaly types. These report types will be reflected in the template selection phase of the report generation that will be further explained in Section 5.7.

There are additional functional requirements that need to be met, which are detected from the initial questionnaire and informal interviews with my supervisor:

- The timestamp of all entries should be provided.
- Technical attributes should be in Dutch.
- Report generation must be able to account for diverse anomaly types and varying amounts of data relevant to an anomalous incident.
- Reports should be able to explain information with a combination of textual and graphical information.

Additionally, we will build on the previous finding of Urumovska [58], which is that the domain experts preferred a more detailed representation of the system that included all state attributes possible. This results in the requirement.

- Technical attributes and state names should be included in reports.

4.3.2. Non-functional Requirements

To ensure the clarity of the reports, some non-functional requirements should be met.

- Sequential relationship of log entries needs to be clearly projected.
- The reports need to be readable and understandable by domain experts.
- The origin of data needs to be clearly projected.
- All routes and elements need to be clearly identified by the standard known by domain experts.

All the listed requirements are general according to the trends of the finding of the initial questionnaire and the goal of the system according to the CGI thesis supervisor. We are not trying to account for the initial preferences of any specific expert group.

5

Report Generation Methodology

This chapter explains the methodology used to design and implement a report generation system for explaining events and anomalies in a train control system from log files. The objective of the methodology is to generate three report presentation versions, each designed to address a specific characteristic of the data. The three versions are the following:

1. *Text*, a purely textual presentation,
2. *TimeText*, a textual presentation with graphical components containing timelines,
3. *RailText*, a finally textual presentation with an annotated railway overview.

In this chapter, we will explain how we process raw log files into three types of final reports. First, we will look into an overview of the designed pipeline for generating reports of all three types from raw log file data in Section 5.1. Subsequently, we will go through each of the pipeline components. In Section 5.2 the steps of the log analysis are introduced, where we end with the methodology to detect anomalous cases. Section 5.3 covers how we extract and interpret data. Section 5.4 then details how the data is organized to be represented in the reports. In Section 5.5 we will go through the steps specific for text. All graphical specific parts are explained in Section 5.6. Section 5.7 covers the design of our Case-Based Reasoning (CBR) system which allows for a dynamic choice of report presentation version for anomalous cases depending on their properties.

5.1. Pipeline

The pipeline architecture that is designed for this project is heavily built on the pipeline that Mahamood et al. [31] introduced, where annotated graphs are generated along with text using the NLG Pipeline architecture. That suits our project well since we aim to include annotated graphs and figures in our generated reports. There, specific actions taken within the implementation of the pipeline are not much discussed in the published paper. The names of their pipeline parts therefore need to serve as the key indicator of what each part entails. They build their pipeline on the foundation of the pipeline introduced by Reiter [40] for data-to-text NLG. That NLG pipeline was discussed in Section 2.2.2 and can be found in Figure 2.1.

Our pipeline can be found in Figure 5.1. It has some major differences in comparison to the original pipeline from [31]. In [31] the data being processed is expected to be unstructured and numerical and therefore the data analysis is structured for data that fits these characteristics. Since we are working from log files that have some structure and are mainly textual instead of numerical, we need to analyze differently. Therefore, the first step was changed from “Data Analysis” to “Log Analysis”.

Instead of annotating graphs like is done in [31], we will make different combinations of text and graphics, sometimes the graphics will need to be annotated by textual information to make the explanations clearer and sometimes the text will accompany the graphics. Since we will create different graphical components and later on will evaluate which component is preferred, we skip the “Visualization planning” component that can be found in the original pipeline. Since its function there is to determine what kind of graphical presentation is relevant for each generated report.

In the subsequent sections, we will describe each pipeline component in more detail.

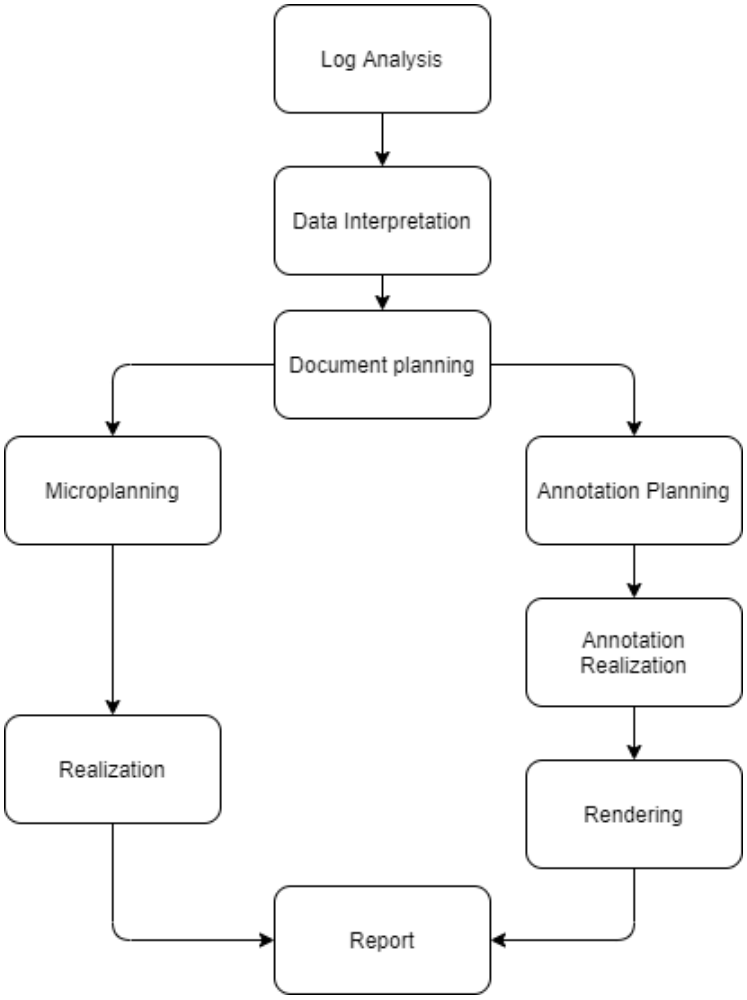


Figure 5.1: Pipeline for report generation

5.2. Log Analysis

The goal of the log analysis phase is to have processed the raw log data so that it is better organized and suitable for querying and other operations. Furthermore, the detection of anomalous cases derived from the log data should be possible and performed.

The core part of the Log Analysis phase is the extensive data processing that needs to be done to convert the log files to a format convenient to work with, match, query and manipulate. The key challenge is related to the sizeable amount of log entries that are generated each day. Many of these entries are not important for the project, so it is vital that the significant entries can be accessed easily. We will now go over what each data processing step entails.

5.2.1. Log formatting

The ARC, IDCR, IDCE and BevNL raw log file types, that were introduced in Section 3.3.1, are structured using XML. However, the structure is not always correct, so some formatting needs to be done for them to be considered as valid and usable. We have to remove excess structural characters and make sure that the files are correctly initialized and opened as XML envelopes. The cause of this fault is the temporal structure of the creation of log files. When one log file is full, the next one is created, so one log file comes sequentially after another. Together they form a time sequence that includes all entries starting from the initial timestamp of the first file to the last timestamp of the latest log file. When a new log file replaces another, it is not treated as a new file, with new initialization of the log file as an XML envelope, but simply as the data is still being written to the former log file.

The raw Meldingen log files, unlike the aforementioned log files, are not structured using XML.

Instead, each entry is represented with an unformatted textual line. Thus they require no formatting in this step.

5.2.2. Log Processing and Parsing

When all log files are in a valid XML format each log file is parsed using Python and imported to Elasticsearch. Elasticsearch is chosen as the storage technology for this project since it specializes in fast querying and good analytical properties. According to Elastic, it is a distributed, RESTful search- and analytics engine capable of addressing a growing number of use cases. CGI has used Elastic Stack in relation to the ASTRIS log files. Elasticsearch is, according to themselves, the heart of the Elastic Stack, it centrally stores your data for lightning-fast search, fine-tuned relevancy, and powerful analytics that scale with ease.[1]

The data structures used by Elasticsearch are called indices. They are comparable to tables in typical database structures. An index has multiple documents, where in our case each document principally represents one log entry, with some exceptions. Each document has numerous data fields and their corresponding values. We import each log file type to a separate Elasticsearch index. We will refer to them as the ARC index, IDCR index, IDCE index, BevNL index and Meldingen index.

In this data extraction step, a lot of foundation work needs to be performed since the structure and keys of log entries are different depending on the log entry type, command type, element type, sender and other properties. Therefore, the design of the parsing schema, to map data from XML to data structure imported to Elasticsearch, took more time than expected. A deep understanding of the log file data was needed to perform this step. The goal is to map a number of diverse log entries to a systematic uniform data structure. For that purpose, we simplify the data structure by converting data from an XML tree structure to a one-level structure with simple key-value pairs. This is all needed to make data-querying possible and efficient.

This is not a straightforward task, since the structure of the entries is complicated and diverse. The general rule is that the XML parent nodes in the body of the XML envelopes represent the action being performed, stating whether it is a request or a state update. The lower level child nodes represent the detailed information that is included in the action. The end nodes, or leaves, of the XML tree, can have different meanings depending on the node structure leading to the end node. Reservation information for elements is a good example; the same 'Gereserveerd' key can refer to different entities depending on the keys of the parent nodes leading to the leaf. Moreover, the key can occur more than once in IDCE entries and have separate meanings each time. In contrast, we have values that have different key values but should be mapped to the same index field. Therefore we need to cast multiple related data keys found in the log entries to one data field in the indices for some log entry keys.

The *Meldingen* log files contain unstructured text as was mentioned in the previous section. Each line in the log files represents an entry. The entries hold the alerts that the system produces. They are not split into keys and values. Therefore, the most common types of alerts had to be investigated to find out what their information entails. The entries are processed according to what kind of error or alert they concern. The processing splits the entries into values which are then inserted into the Meldingen Elasticsearch index.

5.2.3. Detecting anomalous incident cases to explain in reports

Now that the log event data has been imported to Elasticsearch, the data can be scanned for anomalies of types 1 to 4 defined in 4.2.

For this task, we apply misuse detection, which is a technique to detect known anomalous patterns, typically used for detecting security breaches in software systems. As was discussed in Section 2.5.3 in Chapter *Related work and background*, misuse detection is a type of anomaly detection technique where anomalous processed are defined. Matches to these processes are then considered anomalous behavior. We scan the imported log data for matches to these anomalous processes to detect anomalous incidents that should be further interpreted and explained.

Note that the following process profiles use the definition of anomaly types from Section 4.2.

i **ASTRIS Component crashes or stops**

This is a process profile that indicates that ***anomaly type 1*** has occurred. When a software component within ASTRIS crashes or stops, that is projected through the Meldingen log files

[1] www.elastic.co/elasticsearch/

as an error notification. An Elasticsearch query, searching for errors concerning components stopping, allows for a check whether this anomalous event has occurred or not.

ii **The underlying safety layer rejects a request**

This is a process profile that indicates that ***anomaly type 2*** has occurred. The safety layer rejects requests when it deems it such that the outcome of the request is not possible or that is will harm the system. That BevNL rejects a request, means that a request got through the system checks in ASTRIS, that are designed to fulfill all requirements of the safety layer.

iii **Anomalous route status**

This is a process profile that indicates that ***anomaly type 3*** has occurred. The state transitions of a route follow a large complex state machine. In a former thesis on the system [58] this state machine was the sole focus for anomaly detection. In the project, graphical anomaly detection was used to detect deviations from the planned behavior. We use a different way to do this. According to the CGI supervisor of this project, most deviations can be found by searching for a specific set of attributes in a log entry. We search for two specific attributes or an attribute combination here. The attribute combination implies that the preparation phase of the route command has failed and that is stuck in a reservation phase and will never get to the correct state during this process. The other attribute that implies unexpected behavior is when the route gets the status *ingestelopdracht afgekeurd door BevNL* which means that the route command got rejected by the underlying safety layer.

iv **Route request gets rejected by ASTRIS**

This is a process profile that indicates that ***anomaly type 3*** has occurred. All route requests should follow the following route request states, *Controleren, Uitvoeren, Gereed* where *Gereed* should be accompanied with an Acceptatie attribute as 'Geaccepteerd'. When this is not the case the Acceptatie value is 'NietGeaccepteerd'. That means that ASTRIS did not approve of this route request and therefore rejected it. This does not always imply anomalous behavior but is however an unexpected behavior.

v **Anomalous IDCE entries**

This is a process profile that signals that ***anomaly type 4*** has occurred. Here we search for specific values of three attributes in IDCE entries. They are *NietInControle as true*, which indicates that the element is stuck, *Gestoord as true* which indicates that the element is not trusted and *Betrouwbaar as false*, which indicates that the element data is unreliable, i.e. that is could be wrong.

If a match for any of these defined processes is detected, we have identified an anomalous case that should be explained according to our requirements. The exact time of the log entry that projects the anomaly is extracted. The time frame of interest to explain the incident is defined as 4 minutes prior, until that extracted time. The span of this time frame was defined in the requirements for the anomalies in Section 4.3.1.

5.3. Data interpretation

At this point in the pipeline, we have identified a set of anomalous cases that should be explained within the set of log files being processed.

The goal of the Data interpretation phase is to have extracted the set of log entries that are relevant to each anomalous case. To extract all entries relevant to a specific anomalous event, we need to link the different types of log entries together based on the relevant relation for all entries. We will interpret the extracted data to identify aspects that should be included in the reports resulting from our system pipeline.

5.3.1. Extraction of relevant entries

Now the log data has been efficiently imported to Elasticsearch, the next step is to extract all log entries that are related to each anomalous case. Those entries should be related entries within the time frame of 4 minutes before the anomaly as was specified in 4.3.1. The goal of this process is to end up

with a complete set of entries for each anomalous case, that includes all entries needed for the report creation. Here we will go in detail regarding how the entries are linked together and what is included in the extracted log entry set. In this section, a *query* represents a query to an Elasticsearch index, implemented through the official Python package for Elasticsearch [2]. The aforementioned time frame of four minutes is a constant parameter that applies to all queries.

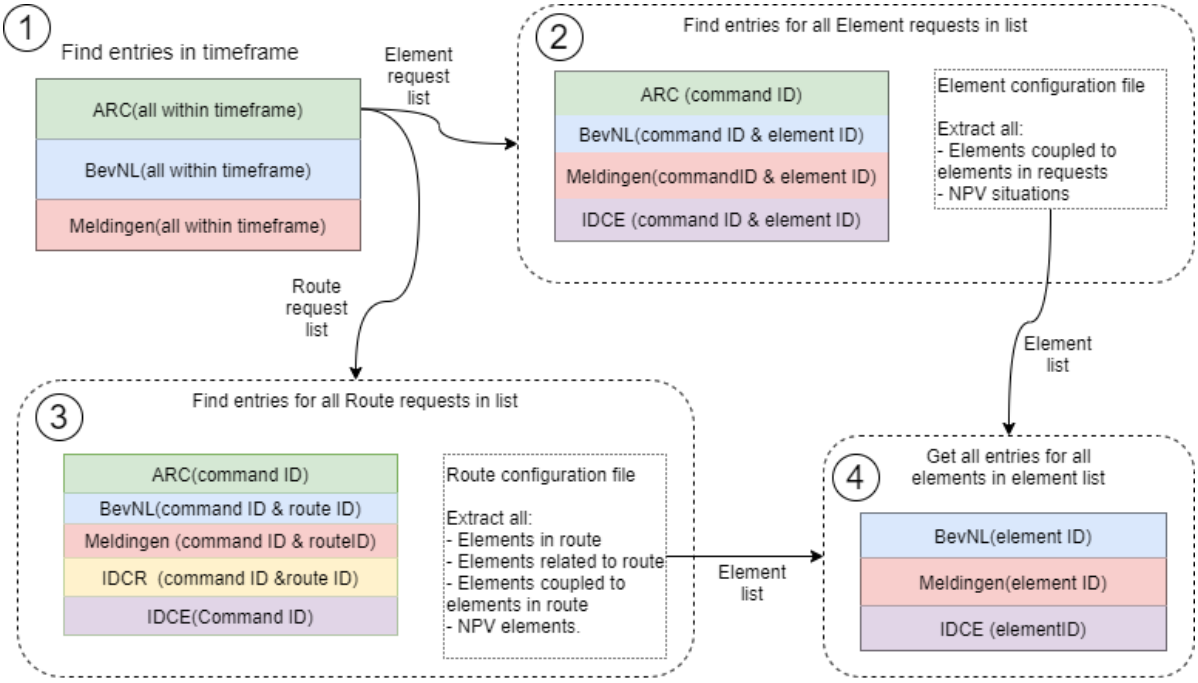


Figure 5.2: Overview of data querying, the numbers refer to the data extraction steps explained in text. The text within the brackets represents the matching parameters used in the queries.

The relationship between different entry types does not always depend on the same linking parameters. In Figure 5.2 we can see an overview of the process of extracting all entries for a specific time frame. The data querying and linking methods are described in four steps:

1. The starting point of extracting entries for a given time frame to find generic entries that apply to the entire time frame and give an overview of the active processes. All ARC entries that occur during a time frame are queried, they hold all core requests that are sent to, and initiated within ASTRIS. Therefore, all requests that can be found in this set of ARC entries are investigated further. The requests are divided into Element Requests (step 2) and Route Requests (step 3). We will go through the functionality that extracts all relevant entries for the request in the following paragraphs. All errors and notifications for the time frame are extracted from the Meldingen Elasticsearch index and the same goes for the BevNL entries showing the communication between ASTRIS and the underlying safety layer. Some BevNL entries are omitted since they are part of so-called “snapshots” which is a mechanism that regularly sends the current status of all elements to check if the system is in the assumed state. That is not connected to any specific request, so they are not relevant for our log entry sets.

2. For all element requests that have been found in the ARC entries from step 1, we go through the following procedure. All ARC, BevNL, Meldingen and IDCE entries that belong to the request in question are queried. The data mapping used here is both the command ID and the element ID. The element ID comprises a combination of five data fields named, *ObjectNaam, ObjectType, GebiedNaam, GebiedType and Infraversie*. The first three are most important, they represent the name of the element, the type of the element and area name where the element is situated. We require a lookup to the element configuration file for the relevant area. There we can find

[2]https://www.elastic.co/guide/en/elasticsearch/client/python-api/current/index.html

information if the element of interest has any coupled elements or if it is part of an NPV situation. The element list is then handled as is described in step 4.

3. For all route requests that have been found in the ARC entries from step 1, we go through the following procedure. All ARC, BevNL, Meldingen and IDCR entries that belong to the request in question are queries. The data mapping used here is both the command ID of the route request being processed and the identification of the route itself. There are many versions of underlying safety layers that ASTRIS communicates with. The different versions use different methods to identify and link entries together, which causes a diversity in how the routes are identified. That is done in two ways, by its begin sign, end sign and LR-string or only by its begin sign and LR-string. Therefore, both identification types need to be accounted for in the data mapping. Now we need to query all IDCE that have a reference to the command ID of interest. The route identification in entries does not contain information regarding what elements are part of the group. From the route configuration file, we extract a list of elements in the route, related to the route (Dutch: Bij rijweg), coupled to or that are part of the same NPV situations as the previously found elements. The found element list is then handled as is described in step 4.

4. Here we describe what entries need to be extracted for each element that has been detected in relation to an element or route request. At this point in the log entry retrieval process, we have already queried all entries that depend on either command or route IDs so we purely query in terms of the identification of the elements. An element is identified with a combination of five data fields. They are *ObjectNaam, ObjectType, GebiedNaam, GebiedType and Infraversie*, like was previously described in step 2. For each element, all BevNL, Meldingen and IDCE entries that match the element ID are queried.

All entries extracted in the four steps described here above are stored in a JSON structure where entries that belong to each case form each file. The JSON structure is analogous to the structure explained in Figure 5.2.

For all IDCE entries, a data cleaning step is performed before the entries are saved to the JSON data file. ASTRIS is required to keep multiple workstations updated on the current position of elements, which results in all status updates being found in multiple copies in the log files. In those copies, a few system values are different between entries, such as the values that describe the recipient of each status update. Nonetheless, the copies are duplicates that must be removed so as not to include duplicate, unnecessary data.

The extracted set of entries for each case is now ready and in a structured form where log entries are separated in sub-parts of the file according to which request they belong to.

5.3.2. Misuse Detection

After extracting the set of log entries that relate to a case of interest, we need to interpret the entries, explore relationships and find out what information contained within them is important to convey in the explanatory reports. Within the log entry set, we group requests depending on what route or element they apply to.

The initial idea was to represent the correct behavior of the system with a finite state machine. A part of the system has been represented by a state machine and therefore it was considered to be a viable solution to represent the complete system. After having spent a considerable amount of time getting to know the system data, the domain and the fundamental properties of anomalies. It was determined in collaboration with the CGI supervisor of the project that a state machine would be too complex. The number of possible state transitions is too high to feasibly handle and not all transitions have even been mapped out by CGI or ProRail. Since it is not in the scope of this project to do so, other ways to detect and map out anomalies were implemented.

In this section we describe the systematic anomaly detection implemented to detect events that are likely to imply anomalous behavior and should therefore be conveyed in the reports generated for each anomalous case. As was done when detecting anomalous cases in Section 5.2.3, we use misuse detection techniques to identify the data that should be included in the reports explaining each case. To start with, we set up profiles of data patterns that represent functionality that can lead to or indicate unexpected or anomalous behavior of the system. The set of entries extracted for each case in Section

5.3.1 is scanned for matches to those profiles. In most anomalous cases multiple factors cause the anomaly and therefore need to be detected to explain the anomaly.

We will explain the defined profiles, see how they match the requirements of the system and how we detect them. We will check for matches to these profiles on a request level. After having searched for matches to all the profiles, each request has a property that says whether its behavior is considered *correct, unfinished or unexpected*. Additionally, we have a property for each request that addresses whether there are *overlapping elements* involved with the request or not. Before we run the misuse detection on a case dataset, we set the behavior as *Correct* and the *overlapping elements* property as false for all requests. If the request behavior does not fall within any of the pre-defined anomalous scenarios we consider them correct.

The first four process profiles are equivalent to the profiles defined earlier for the task of detection anomalous cases in Section 5.2.3. We exclude the first process profile defined for the anomalous case detection since it is not applicable on request level. To clarify the link between the process profiles used here to the ones used in Section 5.2.3 we use the same numbering and skip the first item count.

ii **The underlying safety layer rejects a request** If a request is rejected by the safety layer it is marked with *Unexpected behavior*.

iii **Anomalous route status** Here we query a specific attribute combination in IDCR entries as was described in Section 5.2.3. If those attribute combinations are detected we mark the request with *Unexpected behavior*.

iv **Route request gets rejected by ASTRIS** If a route request does not get accepted within ASTRIS that is informative to know of and mention in the explanatory reports. Therefore, the routes that get rejected this way get marked as *Unexpected behavior*

v **Anomalous IDCE entries** If IDCE entries contain predefined anomalous data values, the requests they apply to are marked as showing *Unexpected behavior*.

In addition to the process profiles directly related to the defined anomaly types, we have defined more process profiles. They detect behavior that was not chosen as the four most important anomaly types to be represented as anomalous incidents in the requirement analysis found in Section 4.2. They can however provide informative details about anomalous incidents if they can be found as a part of anomalous cases. They are therefore sought out and marked accordingly.

vii **ASTRIS rejects a request other than route** This does not always indicate the same anomaly type; it depends on what request is canceled, what anomaly type is at play. The rejected request is however always marked with *Unexpected behavior*.

viii **Mention of route or element in an error** This does not always indicate the same anomaly type, it depends on what request is canceled, what anomaly type is at play. Sometimes errors contain a mention of a specific route or element. When a route ID is mentioned in an error, all relevant route requests should be marked as *Unexpected behavior*. When an element is mentioned the process is a little more complicated since all requests for all routes that are active during the time frame and include the element should be marked as *Unexpected behavior*.

ix **Overlap of NPV situations** If an element is part of two NPV situations at the same time or two requests are making use of the same NPV situation at the same time, that can lead to unexpected behavior. NPV situations were introduced in Section 3.2.2, it is a situation where it is possible that the necessary space around an element to be able to ensure safe movement of trains, overlaps with the space of another element. Therefore, all requests that take part in such overlaps are marked as *Unexpected behavior*.

x **When an element is part of more than one request at the same time** This does not necessarily indicate anomalous behavior, but when elements are part of more than one command it becomes more likely that something will go wrong. For instance, if one command wants to turn a switch to the left, while another requires it to be in the right position. For this purpose, the number of occurrences of each element is counted. Each request gets marked with *No overlapping*

elements if none of its elements are part of another request during the time frame of interest. Otherwise, the request gets the mark *Overlapping elements*. This applies to both route and element requests. Furthermore, each element that is part of a request gets a marker that conveys how many requests make use of the element during the time frame of interest.

When anomalous behavior is detected for a request, the reason for it to be considered as anomalous behavior is stored with the request data. Those reasons are later used when generating the reports to explain why the requests in question are included in the explanatory anomaly reports.

5.4. Document Planning

Working from the interpreted log entry set for all anomalous cases, we now perform document planning with the goal to determine what parts of the extracted data should be included in the generated reports and what should be excluded.

5.4.1. Content determination

Information on all main requests-response pairs that are sent within the system during the time frame of interest should be included in reports since it gives the reader a clear overview of what to expect and look out for in the report. Likewise, all errors that are generated within ASTRIS are a clear indication that something went wrong in the system.

After applying the misuse detection to our set of log entries, all requests should have both the *behavior* and *overlapping elements* properties. Requests that have *Unfinished* or *Unexpected* behavior are important to convey in the explanatory reports. An deviation from this rule regards routes that are unfinished and have no overlapping elements with other requests from the reports, they are excluded in reports since they do not suggest anomalous behavior.

5.4.2. Text structuring and sentence aggregation

The goal of this step is to determine how the data should be organized. That is, how it should be grouped together into sentences and in what order the data should be conveyed. Three reports will be generated for each anomalous case, all three report versions will have the same core arrangement. The length of the document will depend on how many requests are ongoing in the pre-decided time frame and if their behavior is considered correct or not.

Within a report, the data needs to be structured in the order that represents the command flow best, as well as highlighting the information that indicates the anomalous behavior. By doing that we can help the readers of the report to extract the correct understanding from the report. Within a set of entries of the same type, their data should always be ordered temporally, since it is very important in the system that is temporal in nature. The reports will contain both texts in sentences and tables since it differs which text format is better suited depending on the complexity of the data.

To organize the different parts of the reports, we have divided them into smaller parts. The division is very related to the steps of data querying that was explained in Figure 5.2, since both process methodologies are built around the natural data flow within the system. Figure 5.3 shows an overview of the setup of the report. Now we will go through each component conveyed in the structure overview of the report, to understand how the information is organized. We are always referring to entries that apply to the time frame of interest for a specific anomalous case.

Component 1 - Overview This component will serve as an introduction and overview component for each case report. Here the time frame of interest is introduced, along with explanatory sentences to guide the users to understand the content presented in the report. There it is noted that red color will be used to highlight anomalous behavior in the report. All errors are listed in a table format, where some known, serious errors are specifically highlighted. What follows is an overview of all messages sent between ASTRIS and the underlying safety layer. After that, an overview of all requests that enter the system from an operator is presented, where each request is paired with the request's response. In the *TimeText* presentation version, this is followed by graphical components containing timelines, one for all route requests and one for element requests.

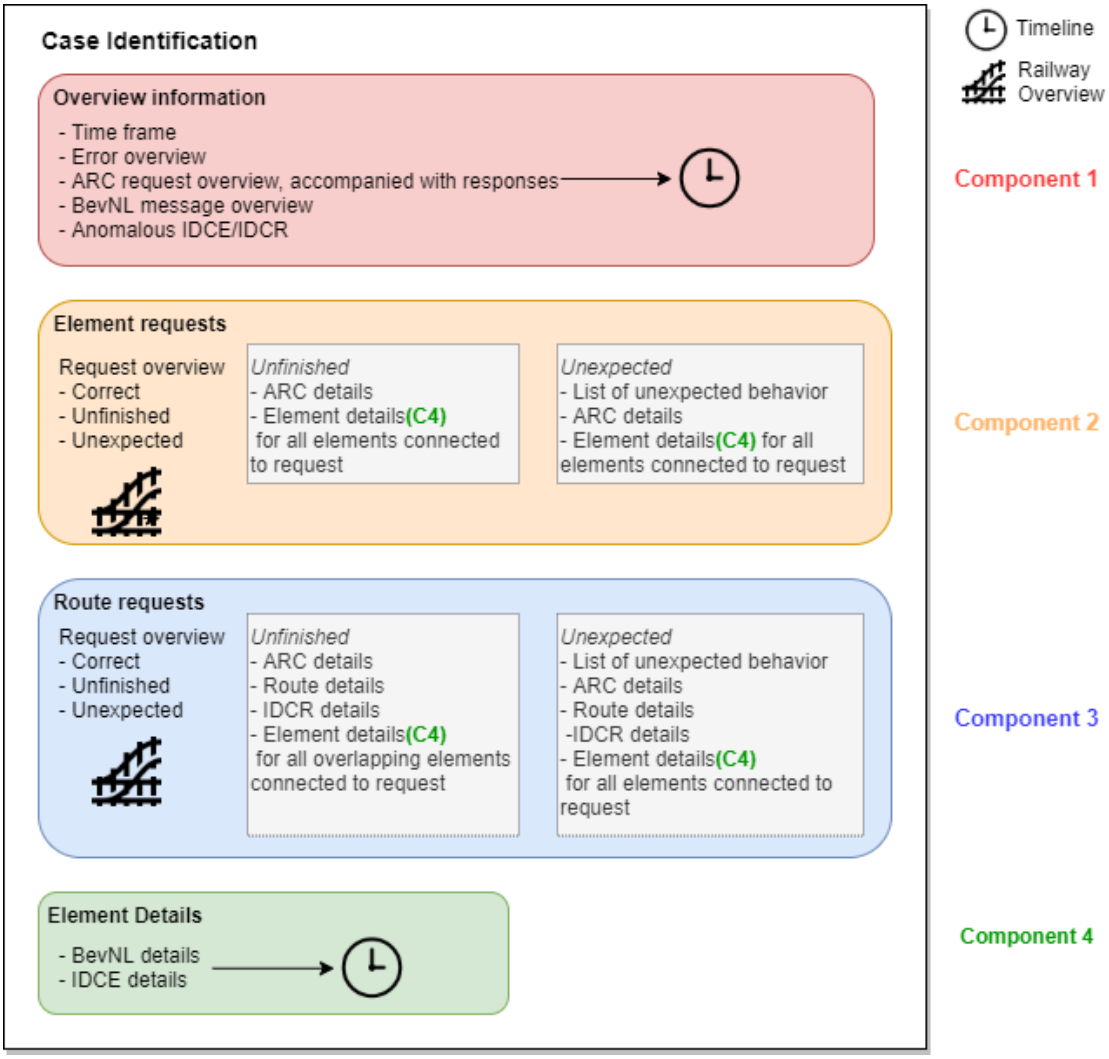


Figure 5.3: Core structure of reports. The timeline icon represents where timelines are inserted in *TimeText* reports and the Railway icon where Railway overview is included in *RailText*. Icons are by *Icons8*

Component 2 - Element Requests Here we start with an overview of the element request being processed during the time frame. The overview lists which element requests are considered correct, unfinished or unexpected. The requests with correct behavior are not handled further. The requests are grouped by the element they apply to. For each element that has some unexpected or unfinished behavior we start with an overview of what behavior was detected that caused the request to fall outside of the *Correct behavior* category. If the element has coupled elements or is part of an NPV situation, that is noted. In the *RailText* presentation, this is followed by a graphical component that contains an annotated railway overview showing the position of the element in question concerning its surrounding elements. Lastly, the element and its related elements are handled as is described in component 4, specified here below.

Component 3 - Route Requests Here we start with an overview of the route request being processed during the time frame. The overview lists which route requests are considered correct, unfinished or unexpected. The requests with correct behavior are not handled further. The requests are grouped by the route they apply to.

This paragraph depicts how each route that has some unexpected or unfinished behavior is handled. We start with an overview of what behavior was detected that caused the request to fall outside of the *Correct behavior* along with configuration details of the route. In the *RailText* presentation, this is followed by a graphical component that contains an annotated railway overview showing the layout of

the route and all elements related to the route. The BevNL log entries are handled next; we list all BevNL entries that relate to the request in question and discuss which requests have been accepted or not. Furthermore, we go through all IDCR entries for all unexpected/unfinished requests for this route. The IDCR entries provide information about the status of the route itself and the status of the commands relating to the route as they are processed within the system. The IDCR entries are represented in a table format since many important attributes need to be conveyed systematically. Finally, the elements in or related to the route are handled as is described in component 4. For requests with *Unexpected* behavior, we explain the behavior of all elements in detail. However, for routes with *Unfinished* behavior we only explain the behavior of elements that overlap with other active routes.

Component 4 - Element Details Here, we go through the entries that show the state updates of all elements that should be handled. Both concerning element- and route requests. The IDCE entries show the state updates from all processes within ASTRIS. The BevNL entries show the state updates as they are in the underlying safety layer. Both entry types are conveyed in a table structure. Following the entry tables, a timeline is included here in the *TimeText* presentation. The timeline shows all BevNL and IDCE entries and the time relation between them.

5.5. Text specific

5.5.1. Microplanning and Realization

In this step, the objective is to find the right words and phrases to express information. That entails selecting words and phrases to identify entities along with referring expression generation which determines pronouns for those entities.

In our domain, many technical terms that bear specific meaning in context to the system, are used by the system experts in daily life. To be able to create an explanatory report that will help them understand certain parts of the system, it is important to make use of the same technical terms as they do. As the initial questionnaire revealed in Section 4.1.2, the experts either want to have technical attributes in Dutch or are neutral whether they are in English or Dutch. To meet the requirements of everyone, all technical attributes will be in Dutch, like they are represented within ASTRIS. Since the system's output is complex and relatively unstructured, it is additionally vital to keep the text simple, understandable and familiar for the experts. Therefore, text choices will be as consistent as possible. A certain piece of information will always be explained in the same way to avoid unnecessary confusion.

For the lower-level planning of text, we make use of templates. As has been discussed in Section 2.1, templates are advantageous when good linguistic rules are not available and for very specific systems[13]. Templates are a good choice for our system since there is no knowledge base available that handles the domain-specific terms of the system. In our templates, the largest part of sentences is pre-defined with placeholders for data specific items in between. Each report is generated from a large dynamic template-like entity that is created with a Python script including domain-specific business logic. We will call this the *main template*. It is composed of smaller component templates, which reflect the components defined in Figure 5.3. Each component template is very dynamic. A component template can be absent or play the biggest role in a report depending on the cases. As well as shrinking and expanding depending on the magnitude of data that needs to be covered within that component template. This structure was chosen since the anomalous cases are diverse and no two cases are likely to be exactly the same. It is not fitting for such a setting to have a specific template for each anomalous case, since that would result in an infeasible amount of templates.

By choosing this dynamic hierarchical template design we can make sure that a report can be generated for all cases that might arise, but of course, their readability and helpfulness might vary depending on the case properties. This is in contrast to the former thesis on the project by Urumovska [58]. There they worked with a static anomaly representation for the finite list of anomalies that were chosen as the scope of the thesis. Now we enhance the scope significantly to handle a much bigger pool of possible anomalies.

There are three *main template* types used in our system, each is used for one of the presentation versions that we have introduced in this chapter.

1. *Text* - textual information, both on table and sentence format.

2. *TimeText* - textual information, both on table and sentence format along with timeline representation of log entries annotated with certain information.

3. *RailText* - textual information, both on table and sentence format along with railway overview annotated with certain information.

We use the same component templates for all anomalous cases. The component templates however contain placeholders that are filled with smaller diverse templates depending on the structure of the cases and their entries.

Now the more detailed component templates will be explained for each of the report components shown in Figure 5.3. Most of the template sentences are static, predefined texts which hold placeholders for data. The placeholders are then filled with the data relevant to each case. As there are many sentences to describe, we will not include syntax trees but explain the sentences. Static sentences will not be explained since no realization is needed for them. We provide examples to explain most sentences. They are indented in the text and in a gray font. Bold parts of the examples represent a placeholder in the sentences. When the placeholder represent a count of some analytical aspect, that will be depicted with an example number. If the placeholder represents a more complex piece of data, the placeholder is depicted with a descriptive textual variable. To realize each template sentence into correctly formed sentences we make use if SimpleNLG which is a surface realizer [3] using a python package. Generated example reports, where we can see an example of the outcome from each component can be found in Appendix B.

Component 1 This report component consists of 6 possible paragraphs, 4 of which are always present. Each paragraph has a distinct purpose to convey an overview of information about processes and anomalies related to one of the log file types. Each paragraph contains some sentences with placeholders for data that needs to be extracted from the set of log entries for the case.

1. The first sentence concerns the introduction of the report. The time frame of the report is given where that start and end time stamps are inserted in a sentence.

   > The time frame of interest for this report is from: **2020-06-18T16:39:30Z** to: **2020-06-18T16:41:10Z**.

2. This paragraph introduces the alerts produced by the system if there are any. Different static sentences are included depending on that. If there are alerts they are presented in a table format, where the timestamp of each alert is inserted in one column and the full alert in the other. The size of this table is determined by the alert count. If there are no alerts, that is explained in a static sentence.

3. The messages and requests sent between ASTRIS and BevNL are then introduced. If there are none, that is reported in a static sentence.

   > **8** BevNL messages were sent between ASTRIS and BevNL during the time period, out of which **3** requests and **3** responses, **3** requests where accepted and **0** declined.

   The following sentence is added if actual status requests are present:

   > Additionally **1** actual status request and **1** actual status responses

4. A similar structure is performed for ARC introduction except it is split up to route and element related requests. The latter sentence parts regarding route and element requests are only included if they are respectively present in the case

   > There were **6** ARC entries within the time frame of interest of which **1** route requests and **1** route related responses, **2** element requests and **2** element related responses.

[3] https://github.com/simplenlg

What follows is a list of details about all requests and their responses. For all accepted and rejected requests we separately insert the following.

> The **3** requests that were **accepted/rejected** are:

We group requests depending on their action and what entity they apply to. Each group is then reported.

> **2 requestName** for **entityID**

Furthermore, the entityID is determined by whether the request group applies to a route or an element.

I For route requests, we have two versions since there are two different route identification protocols within the system.

> route **beginSignal** - **endSignal** - **LRString** / route with begin signal **beginSignal**

II For element requests.

> element **elementType**-**elementID**

5. The next paragraph is only present if there are anomalous IDCR entries within the time frame. In that case, a count of anomalous entries is given in a sentence following a table describing all main attributes of the anomalous IDCR entries

6. This component ends with a paragraph that is only present if there are anomalous IDCE entries within the time frame. Its is similar to the anomalous IDCR paragraph here above, but requires a bit more explanations.

> There are **3** anomalous IDCE entry/ies during this time frame, of which **2** have NietInControl as true, **1** have IsGestoord as true, **0** have Betrouwbaar as false.

This is followed by a table showing the anomalous entries with necessary details.

Component 2 This component consists of 3 parts, where the last one consists of multiple sub-parts.

1. Introduction paragraph explaining the structure of the component following a reported count of element requests that is performed during the time frame.

2. Detailed introduction to all the requests, grouped by their behavior and overlapping element flags found in the misuse detection described in Section 5.3.2. The details for each request is formed as such:

> Element **elementID**-**elementName**-**areaName**with request:**requestAction** at time **requestTimeStamp**.

3. For each element that is being processed in an element request, we go through the details of the behavior related to the element. That detailed overview for each element is further split down into five parts.

I List of all requests that are directly applied to the element. For each of the requests, such a sentence is added to a list after an introductory sentence.

> Request **requestName** with opdrachtID: **requestID** at **requestTimestamp**

II Information whether the element is part of another requests during the time frame is now given. It is done with a static sentence if it not part of any other request. However, if it is part of another request a different sentence is inserted depending on whether the other request is an element or route request. Where the count for each group is included in the relevant sentence.

III If the request has unexpected behavior, all the reasons extracted in Section 5.3.2 and listed here in a sentence format.

IV If the element is part of an NPV situation, that is expressed in a static sentence. If there is no NPV situation, this part is not present in the report.

V The last part is element details from IDCE and BevNL which is handled in component 4 but inserted here in the report structure.

Component 3 The treatment of the route requests is similar to the element requests in Component 2. The first two parts are the same, but now listing up information regarding route requests,instead of element requests, we will therefore not repeat them.

3. The third part of the component considers details regarding the requests similar to the setup for the element requests. We group all requests that have unexpected behavior by the route they apply to. The behavior that regards each route is explained in more sub-parts.

   I All reasons extracted in Section 5.3.2 for the request behavior being considered unexpected are listed. Those reasons explain what anomalies were detected and for which part of the request it applies to. We will not go into detail regarding the setup of these sentences.

   II Information whether there are elements in this route that are also part of other requests during the time frame. Information is depicted with a static sentence when no element is part of any other request. However if any element is part of other requests, a list of all relevant elements is given in a sentence, where the **elementID** and **elementType** are included.

   III Next, a detailed configuration setup of the route is declared in a sentence where seven configuration attributes are reported.

   IV If there are any alerts specified in the initial overview of the report, which apply to the route of interest, they are again projected here in the same format.

   V Now the behavior of each request that applies to the route is covered as follows:

      i. The anomalies concerning the request are listed.

      ii. The details of the ARC entry that holds the request are given

      > The ARC for this entry was at time **ARCtimestamp**, with action **ARCActionName** and opdrachtIdentifiatie **ARCopdrachId**.

      iii. The IDCR entries that regard all status updates for the route are divided into two parts. One part contains IDCR entries that explain the status of the route. The other part contains the status of the request. Both parts are explained in a table format where the most important values are included.

4. The fourth part considers all details regarding all elements that are connected to the route. The elements are grouped whether they are in the route, relevant to the route, coupled to an element related to the route or part of the same NPV situation as some element related to the route.

   I Each element is identified with its 5 identifying data values.

   II Next, a sentence follows that portrays whether the element is part of other processes. A static sentence suffices if the element is only part of one active request. In other cases, it is explained dynamically. The latter part of the sentence, which considers active element request processes for the element, is only included if relevant. Furthermore, the plurality in both parts of the sentence depends on the number of processes.

   > The element is part of **2** active route process(es) and **1** active element process(es) during the time frame.

   III The last part is again element details from IDCE and BevNL which is handled in component 4 but inserted here in the report structure.

Component 4 This component solely handles the details of element state transitions. It is in the same format for element and route requests and is therefore handled as a separate component which is then inserted into components 2 and 3.

1. All IDCE entries are listed up in a table format.
2. All BevNL entries are listed up in a table format.

Both tables are accompanied with a static introductory sentence for each table. The reasoning for using a table format is because many important attributes need to be reported and it needs to be straightforward for the readers to identify when specific attributes change. For that reason, we color table cells yellow when an attribute has changed in that entry compared to the last sequential entry before. Different data values are included in the tables depending on the element type since there are different keys in the elastic indices that hold information relevant to each element type.

Table cells in all aforementioned tables are colored light red if they contain behavior that has been detected as anomalous.

5.6. Graphical specific

The creation of the graphical components is performed similarly for each case. We will split this section into two sub-sections, Section 5.6.1 where we explore the timeline generation and Section 5.6.2 where the generation of railway overview images will be explained. The design of these graphical components has been refined in cooperation with the CGI supervisor of this project to ensure the design is explaining the most important aspects efficiently.

5.6.1. Timelines

The timelines will be generated for especially time-critical and important data which can be difficult to interpret as raw data. It is meant to provide the experts with an uncomplicated way to gain insights into the sequential behavior of the data. We create two types of timelines. One type for handling ARC requests, because of their high importance since they are the core of all processes in the system. For the second type, we will generate timelines for element related entries, both IDCE and BevNL entries combined in one timeline.

Annotation Planning and Realization

The main insights that the timelines are designed to address is to provide quicker insights into the time sequence and context of the entries. The realization will be performed by templates. They are not typical NLG templates since it is not the goal to have the annotations in a sentence format. A different template is used depending on the log entry type. The templates do not represent sentences, just a phrase. A phrase that results from a template application is the text that will later be used to annotate one data point on the timeline. We will first explain annotation planning and realization for the ARC timelines before we explain the element status timelines.

ARC Request timeline For the ARC entries, two timelines can be created for each case. One showing element requests and the other showing route requests. The information that will be included is minimal, namely the name of the request, its time and the response to the request within ASTRIS. Let us begin with the simple template that is used for the ARC timelines. The template can be seen in a graphical format in Figure 5.4. It consists of four data components that are combined in a multi-line text.

1. The number of the request represents its position with regard to other requests that are being processed within the time frame of interest.
2. The type of the request is extracted straight from the ARC entry for the request.
3. The identification type depends on the type of the request. Whether a route or element identification is included depends on whether the request is for a route or an element. The identification is in both cases created from a combination of 2-3 attributes found in the ARC entry for the request.

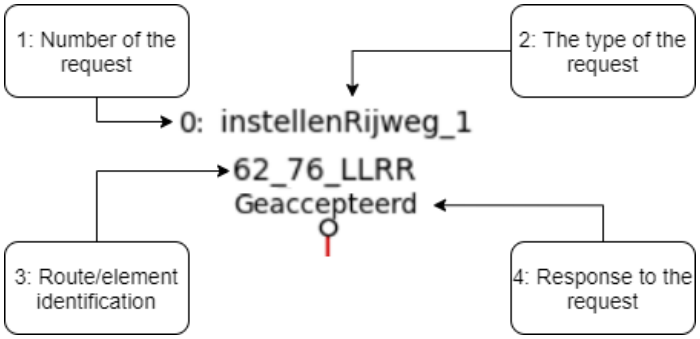


Figure 5.4: Example of template usage for data composition in ARC annotation for ARC timelines

4. To find the response to the entry, a mapping is made on the command ID of the request to find the corresponding response entry from which the response is gotten.

Element status timeline The planning is a bit more complex for the element status timeline in comparison to the ARC request timelines. The log entries containing the element information hold a lot of information and sometimes just a few attributes change between entries. We will emphasize the changing attributes between entries with the timelines.

We aim to visualize all IDCE and BevNL entries for any particular element. The text must not be space-consuming, since that will make the timelines illiterate. The goal is to end up with one timeline for each element that is covered in the report. We do not show all information that is contained within the aforementioned entries; we want to help the readers direct their attention towards attributes that are changing. We have a different template for each of the two different log entry types.

The BevNL entries have a simpler template out of the two log entry types used for the element status timelines. We want to show all data keys that have a changing value between any two sequential entries. For the data keys in question, we then want to include both their keys and value. The template is therefore relatively straightforward and is simply a list of the key-value pairs.

The IDCE entries need a more complex treatment since the data contained in one entry can be immense and very diverse. For this purpose, we make use of a text standard defined for the ASTRIS project. It is called BeheerCLI and is a standard to summarize log entry attributes compactly on a level that the experts are familiar with. The attributes are categorized into 6 categories, element information, Safety layer related information, information on the operation authority of the element, information on the use of the element, information on road obstructions for the element and finally information on obstacles of operation for the element. Each category is then represented with one line according to a specific standard. We will not go into more detail on how this is mapped since that is sensitive information and not necessary to describe for this project. An example of the result from applying this standard can be seen in Figure 5.5

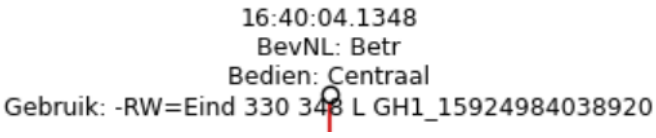


Figure 5.5: Example of log entry information according to BeheerCLI standard

Rendering

In this phase, we insert the generated annotation text to the images. The positioning of the annotation is again slightly different depending on the log type entry. The timelines themselves are created dynamically to fit the number of data entries in each case.

A timeline is composed of timestamps within the time frame of the anomalous case on the x-axis with data points representing log entries on different height levels on the y-axis. The height levels do not indicate different importance, they are purely meant to give each entry sufficient space to prevent

overcrowding the timeline. For the ARC requests timelines, we use 8 different height levels for the data points and for the element timelines we use 5 different heights levels. We see examples of these height levels in Figures 5.6 and 5.7.

We want to make sure there is not too much space between entries on the x-axis. For that purpose, we split the timelines into two parts if there is a time period that spans more than 25% of the active time frame of the element where no entries take place. We can see an example of this behavior in Figure 5.7. When a split is applied, we allocate the width of each timeline part according to the number of entries per each part. That is to ensure that all entries are as evenly distributed as possible with the intention of leaving annotation space for each entry if possible. One timeline can be split more than once if needed.

For the ARC timelines, the annotation is included directly above the corresponding data point. Since the data points are on different height levels, no annotation should overlap while the amount of data entries is suitable. All text for this timeline type is black. An example of an ARC request timeline can be found in Figure 5.6

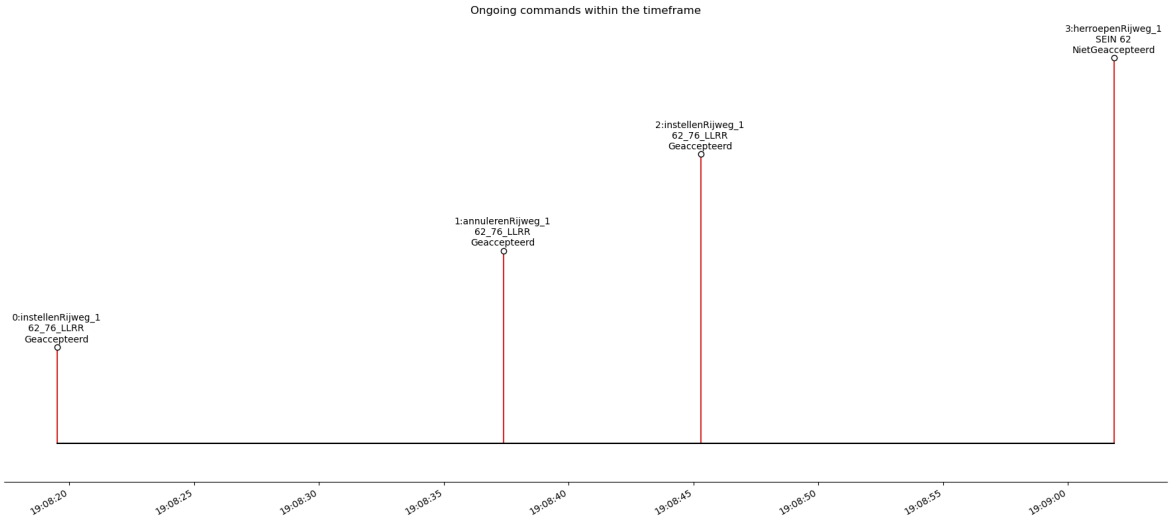


Figure 5.6: Example of a generated timeline for ARC entries

Similarly to the planning procedure, rendering the element status timelines is more complex than the ARC request timelines. First of all, we need to make sure that the distinction between IDCE and BevNL entries is clear. To make a clear distinction between the two different types we make use of different colors to render the text on the timeline canvas. Therefore, a gray color will be used for BevNL entries while black will be the dominant color for the IDCE entries. Furthermore, we will use an orange color to highlight changing values for the IDCE entries. When we have applied the BeheerCLI standard described above, we color the text lines that have any changing values between sequential entries with orange, while the other ones are black. The positioning is done differently for this timeline type when compared to the ARC request timelines since the amount of annotated text can drastically vary. If we were to insert text directly above the data points, there would not be enough space for all annotations. Therefore, the text is inserted around the data point. The exact position is calculated such that if the annotation spans more than three lines, 60% of the lines should be above the height level of the data point and the rest beneath it. An example timeline showing all aspects of the element status timeline can be found in Figure 5.7.

5.6.2. Railway Overview

The railway overview is meant to provide the experts with insights into the context of the data by understanding how the elements in question work together on the actual railway. For the railway, the overview itself is extracted from a train control visualization system called TOON. The system is also developed by CGI and has the main purpose to show the layout of the whole train track system of the Netherlands. For each route or element request, the overview for the area in question is extracted and then annotated with both text and small graphical components. For each extracted overview, the base

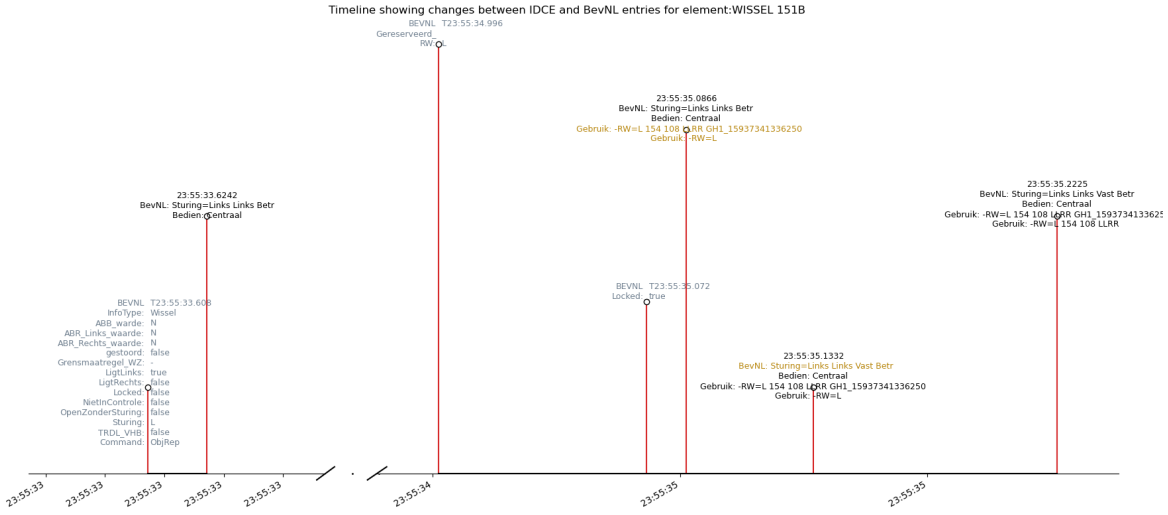


Figure 5.7: Example of a generated timeline for element entries

picture is marked with yellow to signify the route or element of interest. Furthermore, the positioning of all elements is recorded for illustration purposes.

Annotation Planning and Realization
For each element, we split its active time frame into three parts. For the timestamp at the end of each part, a railway overview is annotated. We will refer to this time as the overview time stamp. This split can only be done as long as the active time frame is longer than 3 seconds, which is not always the case, if this is not possible, only one railway overview will be created for the end of the active time frame. For each of the overview time stamps, we want to show the latest status of all entries according to the IDCE entries at the moment of the overview time stamp. We only include the three most common element types in the railway overview, since the rest is not included in the railway illustrations we build our work on. The template for all element types is the same, apart from the choice of attributes selected to appear. They are always the 4 to 5 attributes that give the most insights to the status of the elements. The template setup is best described with an example which can be seen in Figure 5.8. All annotation parts are extracted straight from the IDCE entry of interest apart from the role which is extracted from the route and element configuration files.

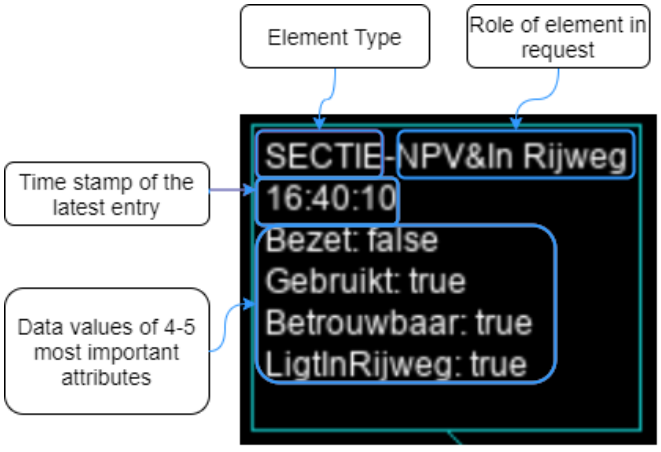


Figure 5.8: Example of template usage for data composition in railway overview annotation

Rendering
In this step, we insert the text generated by the templates to the railway overview. First of all, the time stamp is inserted in the upper left corner of the image of the railway overview. A box and a line

leading to the position of the element identification on the image is drawn. Distinct color is used for this drawing depending on the element type to distinct the types. Next, the annotation text generated for each element is inserted into each corresponding box. An example of a railway overview can be found in Figure 5.9.

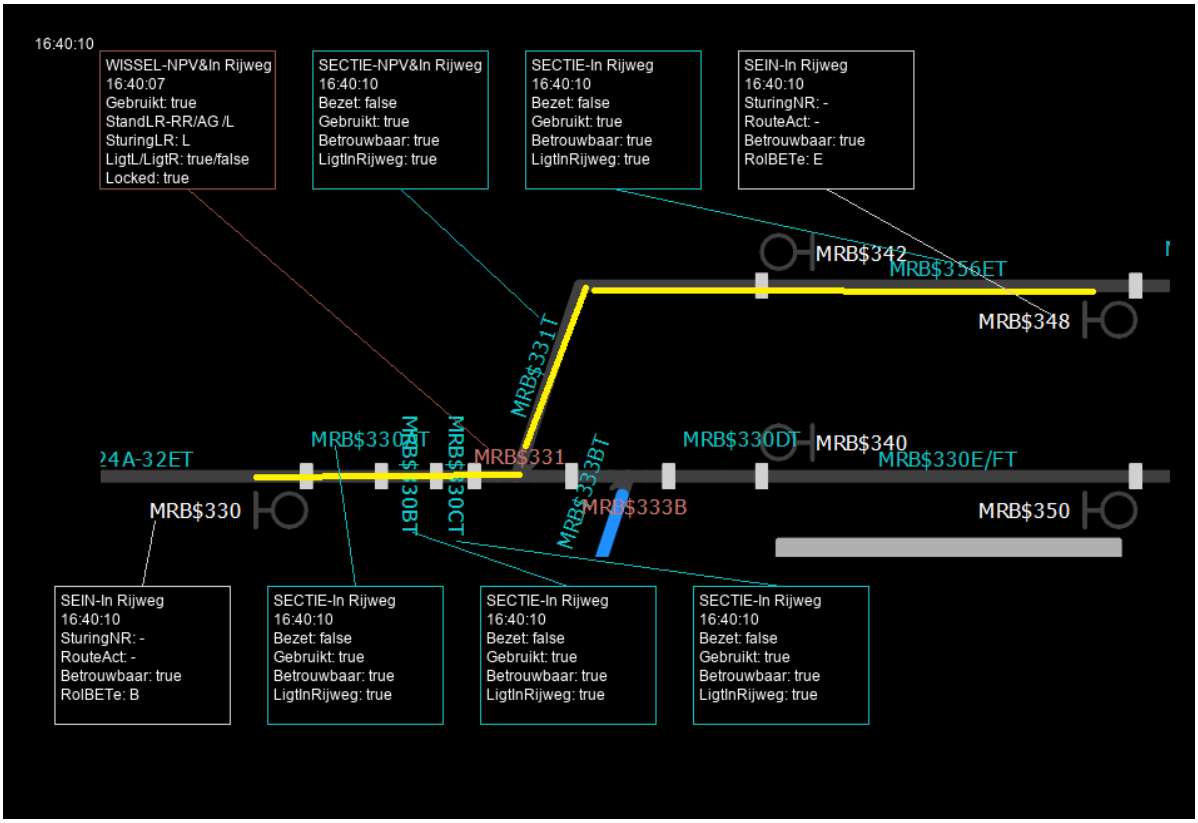


Figure 5.9: Example of a generated railway overview

5.6.3. Combination of textual and visual parts

After having created the textual part of the report for all cases as well as the relevant timelines and railway overviews, the final step of the report generation is to insert the graphical components to the *TimeText* and *RailText* presentations. The components referenced here below are the ones defined in Figure 5.3.

For the *TimeText* reports, the ARC request timelines are inserted in the overview component (Component 1). Furthermore, the element timelines are inserted in the element detail component for each element (Component 4). For the *RailText* reports, the Railway overviews are inserted at the beginning of the relevant request component (Component 2 and 3) following the request overview part of the component. Examples of finished reports containing text and the graphical components can be found in Appendix B.

5.7. Case-Based Reasoning

After all the pipeline has been implemented the three report presentation versions can be generated, We design a simple Case-Based Reasoning (CBR) system. The goal of the CBR system is to predict which of the three report presentation versions, *Text, TimeText or RailText*, is the best for each anomalous case. This is a high-level template selection influenced by the work done in [21] where Case-Based template selection is applied for knowledge-intensive NLG template selection. As has been previously mentioned in Section 2.4, the main advantage of such a system is that we do not need an exhaustive set of solution templates, we can build on similar ones to get the correct outcome. So, in the context of this project, we do not need an exhaustive set of incident-presentation template solutions, but we can

learn from former cases to solve new, unseen cases.

5.7.1. Creation of a Case Base

A CBR system requires a set of solved cases to gain its knowledge from. Since this system does not have any predefined solutions, we need to start by forming a case base. In Chapter 6, two evaluation phases will be performed. We will use the results from evaluation phase 1 to create our case base. 15 diverse cases will be evaluated. We will look into the results from an expert evaluation and combine the findings of multiple questions answered on case level. This will lead to an solution for each case, where each case will be assigned with one of the presentation versions as a solution in the resulting case base. The execution of the creation of the case base will be further discussed in Section 6.2.6.

5.7.2. Methodology

Since the goal of the CBR system is to predict which presentation template is the best for each incident case, it is important to be able to identify distinguishing features in the cases. For that purpose, we introduce 9 properties that are determined for each case. They are all objective characteristics that somewhat describe the events being processed by the TCS during the given time frame. They either concern the magnitude of processes that are active within the time frame or different anomalous properties of the cases. They are the following:

I **Number of Element Requests** is a count of how many element requests were carried out. Categorized into 5 ranges: 0, 1-2, 3-5, 6-9, 10 or more.

II **Number of Route Requests** is a count of how many route requests were carried out. Categorized into 5 ranges: 0, 1-2, 3-5, 6-9, 10 or more.

III **ARC request rejected** is a Boolean property that denotes whether any ARC Request was rejected.

IV **BevNL request Rejected** is a Boolean property that denotes whether any BevNL request was rejected.

V **Route ends up in unexpected state** is a Boolean property that denotes whether any route ended up in an unexpected or anomalous state.

VI **Component Crash** is a Boolean property that denotes whether any component of ASTRIS crashed.

VII **Element Overlap** is a Boolean property that denotes whether any element is part of more than one request.

VIII **Anomalous IDCE** is a Boolean property that denotes whether any element that is part of a request enters a known anomalous state.

IX **NPV overlap** is a Boolean property that denotes whether there is an element that is part of two profile overlap situations.

These properties will be used to represent the cases in the CBR system. A set of values for these 9 properties will be an indication of what the case holds and needs to explain in an explanatory report.

The functionality of a CBR is split into four main components, *Retrieval, Reuse, Revision and Retain* as was discussed in Section 2.4. We will now discuss the design for each of these components in our CBR system.

Retrieval

The presentation selection starts with retrieving a case from the case base. The case that is retrieved is the one that is most similar to a new unseen case looking for a solution. The similarity measure we use is the Mahalanobis distance. All case attributes are taken into account when calculating the distance from the unseen case to the cases in the case base. All attributes have the same initial weight. This is done by applying one-hot encoding to all case attributes. The covariance matrix of the one-hot encoded attributes is then found as well as the inverse of the covariance matrix. The inverse of the covariance

matrix is then used for the similarity calculations to account for possible correlations between case attributes.

The reason for the choice of the Mahalanobis distance instead of other distance measures is because it accounts for the correlation between attributes [26] by using the inverse of the covariance matrix. We considered other frequently used distance metrics such as Euclidean and Manhattan distance metrics. Euclidean does not account for any correlation between features and Manhattan also assumes independent features and is based on the absolute difference value between two data points and works best for numerical values, while our values are more categorical [26]. Literature helped us identify with some assurance that Mahalanobis would be suitable for our data. Such as the empirical study by Khoshgoftaar et al. [26] where a comparison for distance metrics for a CBR system was made between the three metrics and Mahalanobis performed best. Moreover, the CBR design by Cho et al. [11] to detect bankruptcy, found that Mahalanobis outperformed a previous design using Euclidean distance.

To find the most similar case we calculate the Mahalanobis distance using the inverse of the covariance matrix of all properties of the cases. The distance from the unseen case to all the cases that exist in the case base is calculated. The case that has the shortest overall distance from the new unseen case is retrieved.I.e. the case that is most similar to the new unseen case. The solution to that retrieved case is cinsidered as the most suitable solution for the new case.

Reuse
The reuse component uses the presentation version found by the retrieval component and generates the appropriate report version according to the steps we have outlined using the retrieved presentation template.

Revision
The revision part for our system is done by showing the reports to an expert of the system. Like was discussed in Section 2.4, this can be done by an automatic evaluation that uses system-specific rules or expert evaluation. Our system makes use of an expert evaluator. The expert is asked whether it is agreeable that the found solution is the best presentation format to explain the incident case in question. If not, the expert has the power to revise the presentation format chosen for the incident case.

Retain
The retention component serves the purpose of adding new case-solution pairs to the case base when they have been created. This is done after the revision so the solution that is retained to the case base is approved by an expert evaluator before that. By performing this revision, the system manages to gain more knowledge after each case-solution pair it processes. So, when the next new case is injected into the CBR system, it should have increased its ability to make a correct prediction of the best solution for the new case.

5.8. Methodology summary

In this chapter, we have introduced the methodology needed to create explanatory reports from raw log files. We now have an understanding of all the necessary steps to take and what they entail. Furthermore, the difference between the three report presentation versions has been explained.

We are now equipped with all the necessary tools to perform a case study where we create explanatory reports for a set of cases and evaluate them. Furthermore, we will apply our CBR system and investigate if it manages to predict the report version that is most preferred by the experts for each case.

6

Evaluation

After implementing the report generation pipeline, the system can generate three reports per each incident case, each with a different presentation format. The next step of the project is to evaluate the quality of the reports and see which presentation version is most suitable for each incident case. Let us revisit our research question and its sub-questions:

> *When generating anomalous incident reports in a train control system, how should presentation formats be adaptively selected for incidents to provide the highest perceived information quality for domain experts?*

1. What report presentation performs best in terms of perceived information quality and how does it depend on the properties of the anomalous incident?
2. To what extent do expert preferences of report presentations associate with their expert profiles with respect to data familiarity and work experience?
3. How is Case-Based Reasoning, as an adaptive template selection method for anomalous incidents, where incidents are represented with variables, perceived by experts?

The evaluation of the system is split into two phases, phase 1 and phase 2. Each phase is performed with web-based evaluation questionnaires completed by the experts of the system and evaluated as a set of TCS incident cases.

Phase 1 will provide insights to answer research sub-questions 1 and 2. In phase 1, we will evaluate the perceived preference and the perceived understanding of the reports and look into the impact of expert data familiarity and work experience on the experts' opinions. The reason for splitting the evaluation into two phases is that for a CBR system, a case base must be created. Findings from phase 1 will therefore be used for the creation of a case base. The created case base will then be the foundation of the CBR system which will attempt to predict the most suitable solutions for the new unseen experimental cases which will be in the focus of phase 2. Phase 2 will give us insights into the performance of the CBR system. This helps us answer research sub-question 3, where we investigate the performance of the CBR system by measuring the understanding provided and usefulness of the resulting reports generated using the templates chosen by the CBR system.

As has been previously discussed in Section 2.6, there are many possible metrics to measure and many possible ways to measure metrics when evaluating a system that automatically generates text. The most commonly used evaluation technique is automatic metrical evaluation as was discussed in Section 2.6.1. This technique is not an option for this project, since there is no gold-standard data available for the domain and data. We will make use of human evaluation methodologies for the evaluation of this system, which are introduced and explained in Section 2.6.2. We perform both extrinsic and intrinsic evaluation. Using these evaluation methods, we will focus on evaluating the metrics that are deemed most important to the project in question.

We will now go through the design of the evaluation process and its execution. In Section 6.1 we introduce the independent variables that apply to both phases. Section 6.2 introduces all details of

evaluation phase 1, such as the dependent variables, the evaluation questions, the hypotheses we draw from them and lastly the analyzed results and the conclusions to be drawn from those results. In Section 6.3 we discuss the creation of a case base using the results from evaluation phase 1. Furthermore, we apply this case base to predict the most appropriate presentation version for the incident case set that will be handled in evaluation phase 2. Following that, we discuss all aspects such as evaluation questions and results of phase 2 in Section 6.4.

6.1. Independent variables

The main interest and the independent variables that are most important for us are the three **report presentation versions**. In this evaluation phase, we are evaluating the three versions of reports per each incident case. Each report version is generated from the same set of extracted logs, contains the same textual information but the presence of graphical components varies. The three report presentations are:

- **Text**, textual information and no graphical components are included.
- **TimeText**, textual information and graphical components on timeline format.
- **RailText**, textual information and graphical components on railway overview format.

6.1.1. Covariates

There are many more independent variables to acknowledge in the system. They are covariates, that is they represent characteristics of the experimental cases. They mostly relate to the different properties of the incident cases that identify each of them. There are 15 different incident cases evaluated in evaluation Phase 1 and 14 cases in evaluations Phase 2. One of the original cases for Phase 2 was defected so it resulted in 14 cases. Each case has a specified anomaly type, as well as different case properties. The **anomaly types** in question are the same as were defined earlier in Section 4.2, they are when: follows:

1. When an ASTRIS component stops/crashes unexpectedly.
2. When a command From ASTRIS if rejected by BevNL.
3. When a command to set route gets stuck in the wrong state.
4. When an element has unexpected values, that leads to wrong behavior of the element.

These anomaly types do not explicitly describe all important features of a case. Cases that belong to the same anomaly group can be quite diverse but share a common main feature. Even though they have the same main distinctive property it is important to further categorize their characteristics. The **properties** of the cases are for that purpose. The properties describe the behavior of the system during the time frame of interest for an incident case. The properties are the same as were defined in Section 5.7.2 to distinguish cases from each other. They are:

- Number of element requests
- Number of route requests
- ARC request rejected
- BevNL request rejected
- Route ends up in an unexpected state
- Component crash
- Element overlap
- Anomalous IDCE
- NPV overlap

6.2. Phase 1

In this evaluation phase, we evaluate 15 incident cases that have been processed using the predefined report generation pipeline. For each case three report presentations were generated, namely *Text*, *TimeText* and *RailText*. Each case is evaluated in a within-study design and three distinct responses from different experts are gathered for each case. The number of cases was chosen for a few reasons. First of all, to get a good average overview of possible incident cases. Secondly, to gain insights into how the preferences of reports vary depending on the properties of the case and the expert profile. Finally, to not overwhelm the system experts with to many cases to evaluate. The cases are generated from the CGI testing system for the TCS. We use generated test cases to be able to have diverse, guaranteed and recognizable anomalies.

6.2.1. Dependent variables

To avoid sample bias, the cases are randomly distributed between the evaluation participants. This is done to ensure that no two experts have the same set of cases. Furthermore, we need to avoid order bias, which is done by applying Latin Square design which is encouraged in recent literature, e.g in [59]. So the order of the reports in each batch is decided randomly.

Each expert evaluates three or four cases. We want to ensure that each case is evaluated at least three times. We have more cases than experts available, so three random experts got four cases to evaluate. One deviation from this rule occurred when an evaluation turned out to be invalid and the expert that initially performed that evaluation was unable to repeat it. The case was thus reassigned to another expert.

We ensure that each of the three evaluations per case is performed by a different evaluator. To counteract order bias for each case, report presentation versions are displayed and evaluated in a different order for each evaluator. This order of report presentation versions is preserved for all questions for a specific anomalous case, for each evaluator. Each time a case is evaluated it has a different order of presentation versions.

Following is an overview of the dependent variables that are measured in this research phase. they are all evaluated by expert-based human-evaluation methods.

Perceived Preference We start by measuring the perceived preference of the report presentations. This is measured using two questions.

1. First a ranking formatted question where the evaluators are asked to order the report versions according to their preference: *Please order the reports in decreasing order of which report version you would prefer to use to for this anomaly.*
2. Next the readability of each report version is evaluated separately: *How clear in terms of readability do you think the report is?*

Perceived Understanding Secondly, we measure the perceived understanding of the report presentations. This is measured by asking three questions.

1. First a ranking formatted question where the evaluators are asked to order the report version according to how well they help them understand the anomaly of interest: *Please order the reports in decreasing order of how well the report helps you understand the anomaly.*
2. Another ranking question where the evaluators are asked to order the report version according to how well the report represents the system: *Please order the reports in decreasing order according to how well the report represents the system.*
3. Next the performance of each report version is evaluated separately: *To what extent do you feel this report explains the anomaly?*

Usefulness Lastly, we measure the usefulness of the reports overall by asking whether the experts would prefer to use them instead of raw log files. This is not included to answer our research questions, but to get an indication of whether we are meeting the goal of the project in terms of CGI.

1. *Would you prefer to use any of those reports over raw log files to understand the anomaly scenario of interest?*

6.2.2. Materials

A detailed breakdown of the properties of cases evaluated in phase 1 can be found in Table 6.1. There we see all properties for all cases along with the anomaly type each case represents. First of all, it is important to point out the difference in the size of the cases that belong to each anomaly type. We have four cases of type 1, six of type 2, thee of type 3 and lastly one of type 4. It is therefore clear that the set of cases is not balanced in terms of anomaly types and we have more insights regarding some of the anomaly types than others. Furthermore, all the detailed properties for each case, aside from the request counts, are visualized in Figure 6.1. The figure is a visual aid for us to be able to see how different the distribution of these properties is. Starting at the top of the circle representing each case, we have the "ARC Rejected" property as "ARC". The rest of the properties are arranged clockwise around the ring in the same order as the properties are listed in Table 6.23. By observing this figure we see that here are some trends to be seen in the properties of the cases depending on their anomaly type, which is each depicted with a different color.

Case	Anomaly Type	Num Element Requests	Num Route Requests	ARC Rejected	BevNL Rejected	Unexpected Route state	Component Crash	Element overlap	Anomalous IDCE	NPV overlap
0	-	0	2	false	false	false	false	false	false	false
1	1	0	2	false	true	true	true	false	false	false
2	1	0	2	false	true	true	true	false	false	false
3	1	0	19	false	false	true	true	true	true	true
4	1	0	2	false	false	true	true	true	true	false
5	2	0	4	true	false	true	false	true	false	true
6	2	0	2	false	true	true	false	false	false	false
7	2	2	0	true	true	false	false	false	false	false
8	2	2	0	false	true	false	false	true	false	false
9	2	2	0	false	true	false	false	true	false	true
10	2	0	3	false	true	true	false	true	false	false
11	3	0	3	false	false	true	false	false	false	false
12	3	0	4	false	true	true	false	false	false	false
13	3	0	1	false	true	true	false	false	true	false
14	4	3	0	false	false	false	false	false	true	false

Table 6.1: Overview of experimental cases used in phase 1. We group the cases by anomaly types and highlight values where each property is non-zero or true.

Let us take a quick look at the cases. We see that the combination of properties is very diverse but there are certain trends to be found within each anomaly category. We see from Table 6.1 that *ARC Rejected* is only true for incident case with anomaly type 2. The same can be said for a component crash, which is only to be found within anomaly type 1. Other properties seem to be quite random. By looking better into Figure 6.1, where each anomaly type has its specific color for visual aid, we can see that the same combination of properties can never be found for two different cases in two different anomaly categories.

For each of these cases, we have generated three report versions, *Text*, *TimeText* and *RailText*. However, this was not possible for case 13 since the railway overview was not available for that case setting. Thus, only *Text* and *TimeText* where generated for case 13.

6.2.3. Evaluators

A list of 18 experts was provided by the CGI supervisor of the project. They were all chosen because they form the target user group of the system. They share in common having previous or current work experience with ASTRIS. They have served different roles with regards to the system and therefore have a diverse level of knowledge with regards to the log files. This was the same list as was provided

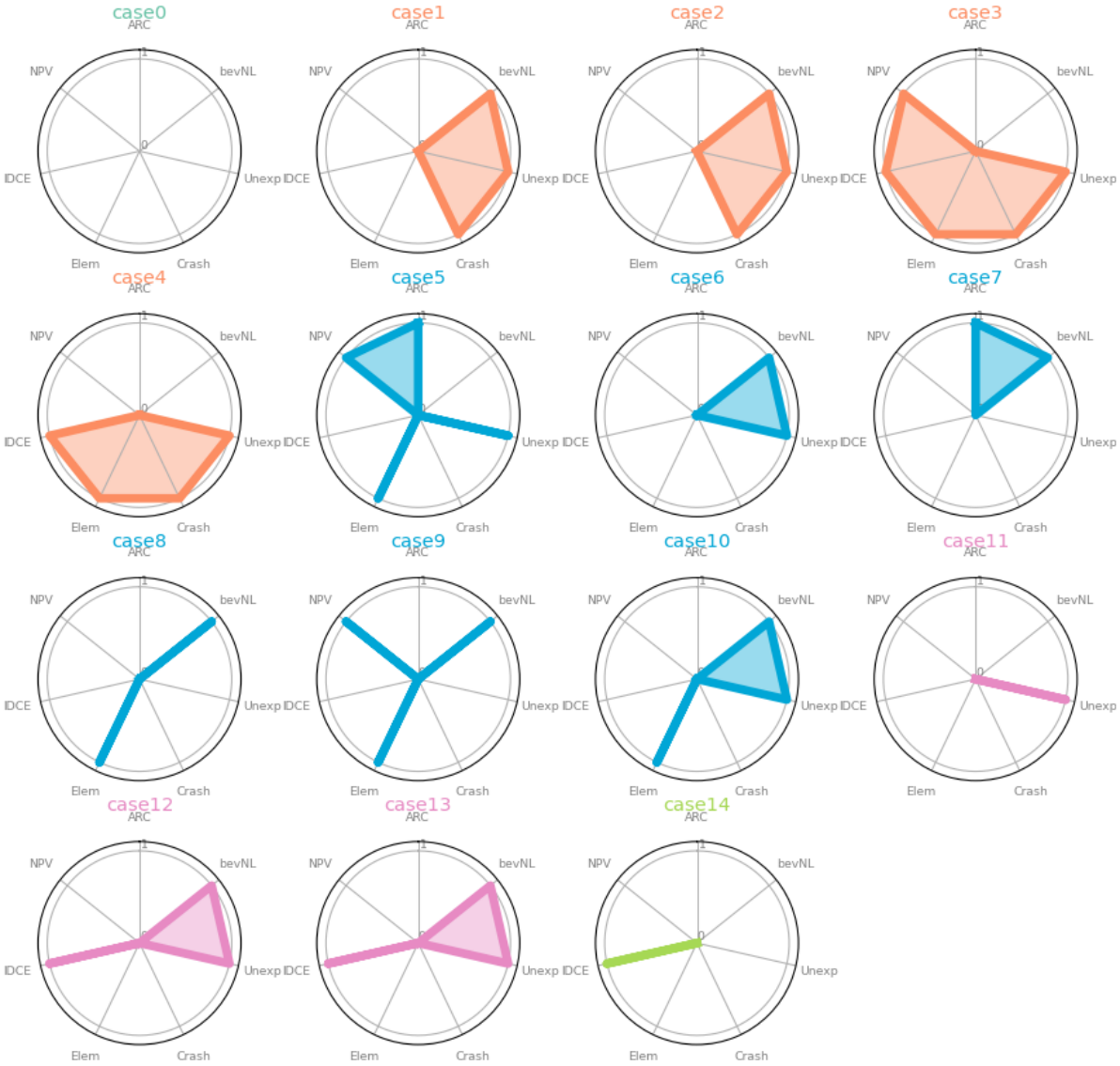
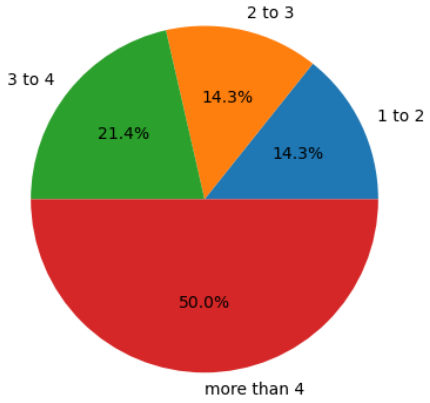


Figure 6.1: Visualisation of case properties for experimental cases in phase 1. We see different anomaly types in distinct colors. This figure aims to show the diverse combination of properties within all anomaly types.

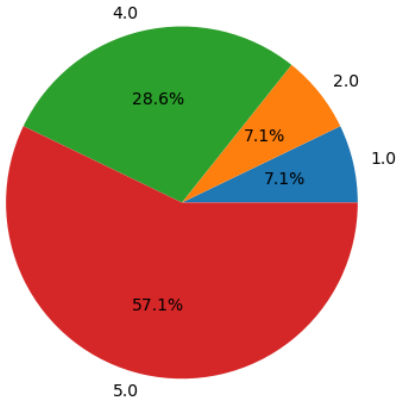
for the initial study. Some experts turned out to be on vacation or unavailable to participate in the given time so the pool resulted in 14 experts. Let us look into the different data we have about these participants.

- 10 work for CGI and 4 for ProRail.
- Seven experts have more than 4 years of work experience, Three have been working around ASTRIS between 3 and 4 years, two experts have an experience between 2 and 3 years and lastly two have an experience of between 1 and 2 years. A breakdown of this information can be seen in Figure 6.2a.
- Furthermore, we know the log file familiarity of the experts on a 5-point rating scale from *Not very familiar* to *Very familiar*. 8 of the experts specified they are very familiar (5 out of 5), 4 experts rated their familiarity 4 out of 5, while one rated 2 out of 5 and lastly one expert is not familiar and gave the lowest familiarity. A breakdown of this information can be seen in Figure 6.2b.
- The experts serve various roles with regards to the TCS. Five are Solution Architects, four are Testers of the system, three Developers of the system, one Product Owners, one Product Manager, one Log file parser and one Architect of a system with an interface to ASTRIS. It is important

to note that three experts specified two roles, therefore the role number of roles does not match up to 14.



(a) Work experience in relation to the system, in years



(b) Familiarity of log files on a 5-point scale from *Not very familiar* to *Very familiar*

Figure 6.2: Expert properties of evaluators of phase 1

6.2.4. Evaluation questions

A within-study was conducted for each case to measure various dependent variables which will help us gain insights to answer our research questions. We will focus on measuring the quality, readability and understandability of the reports like has been discussed in Section 2.6.2. Each evaluator of a case received all presentation versions for each case he evaluated. Each presentation version is evaluated specifically in terms of readability and explanation performance. We then compare the report presentation in terms of perceived preference, perceived understandability and how well they manage to represent the system. The incident case description is given for each case, so that all evaluators have the same information about what incident information the reports is supposed to be explaining.

We make use of a few question types in our evaluation. When the rating scale is appropriate, we make use of a 7-point rating scale. That choice is based on the finding of [59] that the 7-point scale is found to maximize reliability, validity and discrimination. This was discussed in 2.6.2. Seeing that we are comparing three different report versions, we believe that ranking is an appropriate question format for some questions. A ranking question format has not been extensively used but has been receiving some praise recently. We can see an example of ranking questions being used for an NLG system in [50], where different presentation versions of diving data are evaluated.

We will split our evaluation questions according to which metric they measure and we will classify the metrics in line with whether they measure extrinsic or intrinsic aspects of the system. The difference between these two types of evaluation was shortly discussed in Section 2.6.2. To brush up our memory, extrinsic evaluation measures the competence of a system to meet its assigned goal. While intrinsic evaluation focuses on the more general properties of the output of the system, such as readability.

Extrinsic evaluation

We pose multiple questions that evaluate how well the system manages to achieve its main goal, which is to help the system experts to understand anomalous incidents. Those questions count as extrinsic evaluation and are important for our system to measure how well we manage to meet our goal. Extrinsic evaluation needs to be performed by its target users and in its right context, which is exactly the evaluation setting we have set up.

We consider different aspects of meeting our goal and by summarizing them we aim to be able to give an overview of the capability of the system to meet its main requirements.

Firstly, we measure the **performance** in terms of helping the experts understand the anomalous incident that is explained in the reports. For each report version, we ask Question 1.1 that can be seen here below. The experts were asked to rate the reports on a 7-point rating scale on the scale from “Very poorly” to “Very well”. In Figure 6.3 we see the layout used for asking questions on a rating scale.

- **Question 1.1:** *"To what extent do you feel this report explains the anomaly?"*

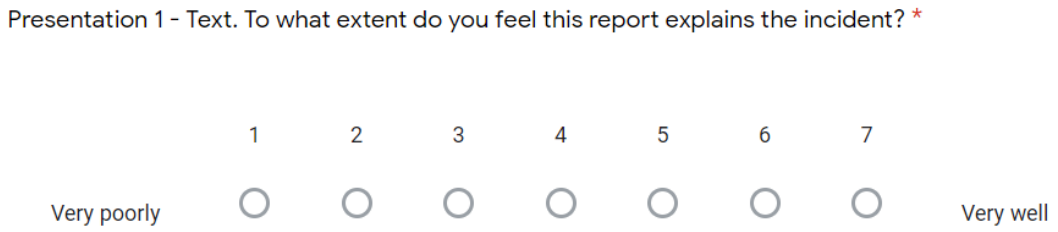


Figure 6.3: Layout of 7-point rating scale questions

We also measure how the reports compare to each other by asking the evaluators to arrange the different versions according to specific questions. The **performance** of explaining the anomalous incident is again measured in Question 1.2. This question measures the same criteria as Question 1.1 but furthermore gives us clear insights into which report version is most informative for the experts.

- **Question 1.2:** *"Please order the reports in decreasing order of how well the report helps you understand the anomaly."*

For the direct ranking questions, which are Question 1.2 - Question 1.4 we use the same format. The experts are required to rank the reports in a decreasing order according to the relevant question. It is mandatory to assign each report version to a separate place. That is, it is not a possibility to set two versions in the same ranking place. In Figure 6.4 we can see the format of the questions.

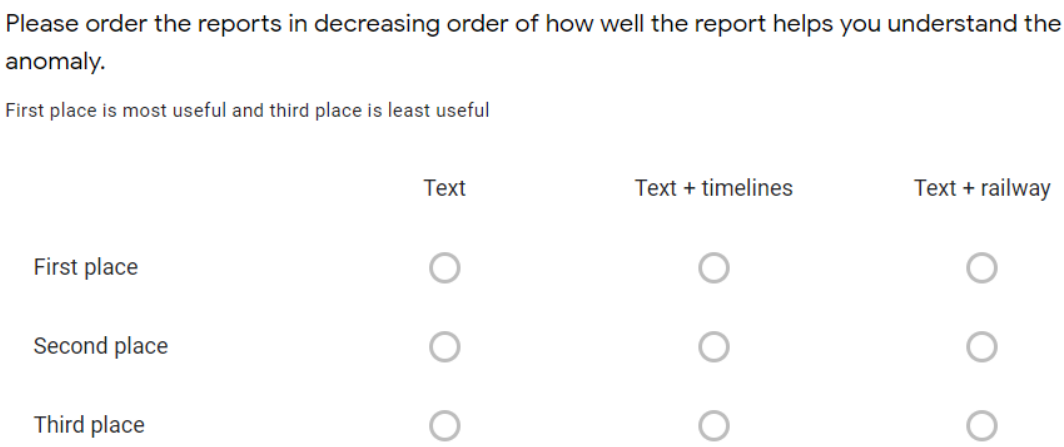


Figure 6.4: Layout of ranking questions

Next, in Question 1.3, we measure which report version the experts have a **preference** to use when working with such an anomaly. The preference was measured since it has been pointed out in other research comparing visual and textual representations that the perceived preference of presentation does not necessarily line up with the understandability of the presentations [29, 39]. Therefore, this question is influenced by the work by Law et al. [29].

- **Question 1.3:** *"Please order the reports in decreasing order of which report version you would prefer to use to for this anomaly."*

Lastly, we wanted to evaluate which report presentation managed to **represent the system** best according to the experts. Again the question format described above is used.

- **Question 1.4:** *"Please order the reports in decreasing order according of how well the report represents the system."*

Intrinsic evaluation
Here we measure a few standard evaluation metrics for NLG systems, namely readability and usefulness.

We measure **readability** of each report version by asking Question 1.5 which can be seen here below. The experts were asked to rate the versions of the report on a 7-point rating scale on the scale from “Very unclear” to “Very clear”.

- **Question 1.5:** *“How clear in terms of readability do you think the report is?”*

Thereafter the **usefulness** of the reports is measured. This is not asked separately per presentation version but applies to all the report versions for a case to get overall insights into the usefulness of the reports. Question 1.6 shows whether the experts deem the reports to be good enough to make use of instead of manually looking at the raw log files. The options given for this question are “Yes”, “No” and “Maybe”.

- **Question 1.6:** *“Would you prefer to use any of those reports over raw log files to understand the anomaly scenario of interest?”*

Free Text Evaluation
Lastly, each expert evaluator was given an open optional question. There we gave the opportunity for open comments or remarks regarding the anomaly case. Here the experts can convey aspects they thought were lacking, good or could be done differently.

- **Question 1.7:** *“Any additional comments or remarks regarding the reports for this anomaly case?”*

6.2.5. Hypotheses
In this evaluation phase we test a few hypotheses:

Hypothesis 1 (H1): *Report version containing graphical components are overall preferred over pure textual reports.*

H1 concerns the preference of the reports and is tested with questions 1.3 and 1.5, both with extrinsic and intrinsic evaluation.

Hypothesis 2 (H2): *Report versions containing graphical components provide a better understanding than pure textual ones.*

H2 concerns the performance and how well the reports manage to achieve the goal of the system. It is tested with questions 1.1, 1.2 and 1.4.

Hypothesis 3 (H3): *The same report presentation is not best suited for all cases.*

H3 concerns the overall suitability of the reports and is tested with a combination of question 1.1 to 1.5.

Hypothesis 4 (H4): *The perceived preference of the experts is associated with the experts’ work experience and their familiarity with the log output of the train control system.*

H4 provides insight into the difference of preference between the experts. A combination of the user model question and questions 1.3 and 1.5 are used to test this hypothesis. This hypothesis is related to H1.

Table 6.2 provides us with an overview of the relationship between all research questions, hypotheses and criteria measured. It is meant to help us navigate through the current chapter.

6.2.6. Execution
The evaluation questionnaire was carried out through an online survey since circumstances due to Covid-19 did not give the possibility for in-person interviews. We made use of Google Forms [1] to perform the evaluation. The questionnaire consists of the following components:

[1] https://www.google.com/forms/about/

Research question	Hypotheses tested	Criteria measured
sub-RQ1 What report presentation performs best in terms of perceived information quality and how does it depend on the properties of the anomalous incident?	**H1** Report version containing graphical components are overall preferred over pure textual reports.	Preference & Readability
	H2 Report versions containing graphical components provide better understanding than pure textual ones.	Performance & Representation of the system
	H3 The same report presentation is not best suited for all cases.	Preference, Readability, Performance & Representation of the system
sub-RQ2 To what extent do expert preferences associate with their expert profiles?	**H4** The perceived preference of the experts is associated with the experts, level of expertise and their familiarity to the log output of the train control system.	Preference & Readability

Table 6.2: Overview of the relationship between research questions, hypotheses and criteria relevant to Phase 1.

Introduction Here we introduce the project and its goal to ensure all participants have the same information. We specify which log files of the system are included in our work. We inform the participants about how many cases they would evaluate and explain that there are three versions of reports per each case. Lastly, participants are asked to answer all questions to the best of their knowledge and as clearly as possible.

Consent Form Next the participants are asked to agree to a consent form approved by the ethics committee. The consent form is important so that we have the confirmed permission to use their answers for research purposes, along with giving the participants information about their rights, e.g. to quit the study at any given time. Furthermore, the participants are asked to insert their names so we can link the collected consent to the participant.

User model Following the gathering of consent, the participants are asked to answer three questions regarding their professional properties. The questions are the same as were used for the user model in the initial survey described in Section 4.1.1. They concern log familiarity, experience with ASTRIS and role in relation to ASTRIS.

Report Evaluation Lastly, the reports are evaluated. For each case, the three report versions are provided through an internet link along with the description of the anomaly that is explained in the reports. Following that the evaluation questions which were described here above in Section 6.2.4 are presented.

6.2.7. Result Analysis Methodology

This section will cover the main analysis methodologies we will use to analyse the results from our evaluation questionnaires.

Ranking Questions

In the ranking questions we ask the experts to rank the three presentation versions, *Text*, *TimeText* and *RailText* according to a specific criteria. The criteria we measure using this format is the performance, preference and the ability of the report presentations to represent the TCS correctly, as was introduced in Section 6.2.4.

A descriptive analysis can be applied to these results, similarly as is done in [50]. We assign 3 points for ranking in first place, 2 points for second place and 1 point for third place. This will result in a total value, mean and standard deviation for each presentation type, that represents how high it was ranked overall. The total values for each presentation can then be compared to give a final ranking over all the presentations.

Statistical analysis methods will be applied for the ranking results as well. The statistical test that we will run to check whether a significant difference is between the behaviour of the different presentation versions is Friedman's test. It is a suitable test when three or more treatments are given to each evaluator, which is relevant to our setup. In our case the treatments are the three report presentation versions. The test is only considered valid if it has five or more data samples to analyse, where each sample has a measurement for all treatments. The null-hypothesis being tested is whether the treatments being compared have the same distributions [43]. We use a significance level of α=0.05 in our analysis.

In the event of a significant difference being confirmed by Friedman's test, all pairwise comparisons are tested as Conover [12] suggests using an extension of the Friedman's test. It is built on the student T- distribution statistic which assumes normal distribution of data.

For two evaluation questions, we will perform a non-parametric Cochran's Q test. The test assumes binary values, which will in our case be represented by whether a presentation version is ranked above the other presentation version in the comparison. That is, when we are evaluating first place, the presentation version that was ranked first will be represented as *1* and the other two as *0* for each evaluation result.

Rating scale

In the rating scale questions we ask the experts to rate each presentation version on a 7-point scale, where 1 represents the worst performance according to the criteria in question and 7 the best. We will test some different criteria using this format as was discussed in Section 6.2.4, such as performance and readability. For these questions we will calculate the mean, median and standard deviations of evaluations for each presentation version for each case. We will also provide the overall mean, median and standard deviation of all evaluations for all cases.

To check whether there is a significant difference between the means for the three different presentations we will perform a one-way ANOVA, which assumes a normal distribution of data. We use a significance level of α=0.05 in this analysis, like for the ranking questions.

Open Questions

In phase 1 we have one open question. It will be analysed with methodology inspired by grounded theory analysis. There we will code the answers on a sentence level to identify main trends and concepts of information. Furthermore we will categorize the concepts and discuss them.

6.2.8. Results

The results we will discuss from this evaluation phase are twofold. First of all, we want to answer the hypotheses we defined for this phase and then we want to determine the best solution for each of the cases to form a case base. That will be done in Section 6.3.

H1: Report versions containing graphical components are overall preferred over pure textual reports

This hypothesis is tested with two evaluation questions. They are question 1.3, which measures the expert's preference of report presentations, and question 1.5, which measures the readability of the reports.

We begin by looking into the answers to question 1.3. They can be found in Table 6.3. The table shows the combined results for the ranking of presentation versions for all cases. Descriptive analysis can be applied to these results, similarly as is done in [50]. We assign 3 points for ranking in the first place, 2 points for second place and 1 point for third place. This results in *RailText* scoring highest. It is also important to point out that the railway presentation was not available for one of the experimental cases, so the chances for the other two presentations were higher than for RailText.

First, we will analyze the overall outcome of this ranking question. In one case out of the 15, the *RailText* version was not available, which resulted in a ranking question between only two presentation

	First place	Second place	Third place	n	Mean rank	Total Value	Ranking
Text	14 (31.1%)	13 (28.9%)	18 (42.9%)	45	2.09	86	2
TimeText	15 (33.3%)	9 (20%)	21 (50%)	45	2.13	84	3
RailText	16 (35.6%)	23 (51.1%)	3 (7.1%)	42	1.69	97	1
n	45	45	42				

Table 6.3: Question 1.3: Ranking question measuring preference of report presentations for each anomalous case. Cumulative results from three evaluations per case performed by 14 experts. The total value is calculated with a descriptive analysis of the ranking data. The n represent the number of evaluations.

Test statistic	Value
N	42
DF	2
chi-square	6.0476
p	0.0467

(a) Friedman's test for cases that had all presentation version available

Pair being tested	p-value
Text - TimeText	0.9112
Text - RailText	0.0365
TimeText - RailText	0.0279

(b) Conover post-hoc test for all presentation pairs

Table 6.4: Question 1.3: Statistical tests

versions for that case. The answers for this anomalous case are not included for the Friedman´s test, to ensure fair treatment for all presentation versions. For this test a first-place ranking is awarded the value *1*, second place is awarded *2* and third place a *3*. By running the Friedman's test on our results we get a p-value of 0.0467. This is slightly less than our significance threshold of 0.05. This indicates that the null-hypothesis of all presentations having the same distribution is rejected as there is some difference in the behavior of the different presentations. Pairwise comparisons are then tested with a Conover post-hoc test to find out where the difference lies. The ranking outcome of the descriptive analysis implies that there is some difference between the performance of *RailText* in comparison with the other presentations, which can be confirmed with the following results. The difference between *Text* and *TimeText* is not significant (Friedman's, p = 0.9112), however there is a significant difference between *RailText* and the other two presentation versions. The comparisons between *RailText* and *TimeText* implies the largest difference with a p-value of 0.0279, the comparison of *Text* and *RailText* results in a p-value of 0.0365 which also is a significant difference. In both significant differences, *RailText* presentation yields better results in comparison to the other presentation version in the test. So overall, the *RailText* presentation is most preferred by the experts.

To take the case that was excluded in Friedman's test into consideration, we can run Cochran's Q test on the pairwise results only between *Text* and *TimeText* (Cochrans's Q =0.556,p = 0.4561). The test tells us there is not a significant difference between the two presentations.

The conclusion to be drawn from the results for question 1.3 is that *RailText* is preferred to the other two presentations.

For question 1.5, we can see the average, standard deviation and median for each report version, as well as the overall outcome score for all cases combined, in Table 6.5. We see that *RailText* has the highest total average and *TimeText* the lowest. The calculated averages for the presentation versions are not significantly different from one another (ANOVA F(2,129) = 1.66, p = 0.1925). So no definitive conclusion can be drawn from these averages regarding which presentation version provides the best overall readability.

Some insights can nevertheless be drawn from the results. We have highlighted the average for the cases that get the highest and lowest average for each presentation type. The purple highlights represent the minimum values achieved. We can see that case 6 gets the lowest evaluation out of all the cases for all presentation versions. Case 3 has an equally low evaluation as case 6 for the *RailText* evaluation and does not perform much better than the minimum for the other two presentations. Case 3 had a drastically large amount of requests compared to other cases being handled, which might explain this behavior. It is interesting to see that the *TimeText* presentation got a lower minimum value than the other two presentations. Now, let us look into the maximum values obtained, *RailText* presentation has

	Text			TimeText			RailText		
Case	**Avg**	**Med**	**Std**	**Avg**	**Med**	**Std**	**Avg**	**Med**	**Std**
0	5.0	5.0	1.0	4.33	4.0	1.53	5.0	5.0	1.0
1	5.0	5.0	1.0	**5.67**	6.0	0.58	5.67	6.0	0.58
2	5.33	5.0	0.58	4.67	5.0	1.53	**6.33**	6.0	0.58
3	3.67	3.0	1.15	2.67	3.0	1.53	**3.33**	3.0	1.53
4	3.67	4.0	2.52	4.0	5.0	2.65	3.67	5.0	2.31
5	3.67	5.0	2.31	3.67	5.0	2.31	3.67	5.0	2.31
6	**3.33**	4.0	2.08	**2.33**	2.0	1.53	**3.33**	4.0	2.08
7	4.67	5.0	0.58	**5.67**	6.0	0.58	5.67	6.0	0.58
8	5.0	5.0	0.0	4.33	4.0	1.53	5.67	6.0	0.58
9	**5.67**	6.0	0.58	4.67	4.0	1.15	6.0	6.0	1.0
10	4.67	5.0	1.53	4.33	4.0	0.58	4.67	5.0	1.53
11	4.33	5.0	1.15	**5.67**	6.0	0.58	5.33	5.0	0.58
12	5.33	5.0	1.53	3.67	4.0	0.58	4.33	5.0	1.15
13	4.67	5.0	1.53	3.67	4.0	1.53	5.33	6.0	1.15
14	**5.67**	6.0	0.58	5.0	5.0	1.0	-	-	-
All	4.64	5.0	1.37	4.29	4.0	1.53	4.86	5.0	1.51

Table 6.5: Question 1.5: Measures the readability of the reports on a 7-point rating scale. Each case has three evaluations, they were performed by 14 out of 18 participants from the expert pool. The lowest case average for each presentation version is colored purple and the highest is colored blue.

the highest maximum of the three presentation versions. It also has the second-highest, which equals the maximum for the other two presentations for three cases. The maximum values for the different presentations do not align to a specific case like the minimum. It is interesting to see that the standard deviations for cases 4,5 and 6 are the highest, while the scores are lowest. This shows that the experts that evaluated these cases did not agree on the evaluation.

Summary H1 We have concluded that *RailText* is overall preferred over pure textual presentation. The same does not apply for the *TimeText* presentation as is it not preferred over pure textual presentation. That can be a result of the dynamic nature of the timelines and the incident cases. That is, the cases have a great amount of data that needs to be conveyed, which can result in the timelines getting more crowded. This can cause the timelines to confuse the reader instead of aiding in his task. Our hypothesis: *"Report versions containing graphical components are overall preferred over pure textual reports."* is therefore not supported. We nevertheless have the conclusion of *RailText* being preferred over purely textual reports. The standard deviations are however large and cause an overlap of evaluation scores for the different presentation types.

H2: Report versions containing graphical components provide a better understanding than pure textual ones

This hypothesis is tested with the help of three evaluation questions. Questions 1.1 and 1.2 measure the performance of the report in terms of how well they manage to explain the anomaly. Question 1.4 measures how well the reports manage to represent the system. Next, the results of each of the evaluation questions will be analyzed.

First, we investigate the results found in Table 6.6 for question 1.1. There the ability to explain the anomaly is measured on a 7-point rating scale. We see that on average the *TimeText* presentation performs best in terms of explaining the anomaly with an average value of 5.0. *Text* has an average of 4.84 and *RailText* an average of 4.71. These average values are calculated over all cases. However, if we look into the total standard deviations of all the cases we see that there is quite a bit of overlap. The significance of these values was tested with an one-way ANOVA test (ANOVA $F(2,129) = 3.05$, $p = 0.85$), which confirms that there is no significant difference between the distribution of evaluation values for different presentation versions. Therefore we can not conclude which presentation best explains anomalies from these results.

Despite that, we can gain some insights. We have again highlighted the lowest average for each presentation with purple, and the highest with blue. The lowest score awarded is 2.67 for all presentation versions. That score is given for case 3 in all cases, which also got rather low evaluation in terms of readability as was discussed in the H1 Section. The highest individual case average can be found for *Text*, even though the highest overall average is for *TimeText*. The standard deviations for all groups are quite high, which indicates that the experts do not agree frequently on their evaluations.

	Text			TimeText			RailText		
Case	**Avg**	**Med**	**Std**	**Avg**	**Med**	**Std**	**Avg**	**Med**	**Std**
0	5.67	6.0	1.53	5.33	5.0	1.53	5.67	6.0	1.53
1	4.67	5.0	0.58	5.67	6.0	0.58	5.33	5.0	0.58
2	5.33	5.0	0.58	5.0	5.0	1.0	5.67	6.0	0.58
3	**2.67**	**2.0**	2.08	**2.67**	**2.0**	2.08	**2.67**	**2.0**	2.08
4	4.0	4.0	3.0	4.67	6.0	3.21	4.33	5.0	3.06
5	3.33	4.0	2.08	3.33	4.0	2.08	3.33	4.0	2.08
6	3.67	4.0	2.52	3.0	2.0	2.65	3.67	4.0	2.52
7	4.33	5.0	1.15	5.33	5.0	0.58	5.33	5.0	0.58
8	5.33	5.0	0.58	5.0	6.0	1.73	5.67	6.0	0.58
9	6.0	6.0	1.0	6.0	7.0	1.73	6.0	6.0	1.0
10	5.67	5.0	1.15	5.67	5.0	1.15	6.0	6.0	1.0
11	5.33	6.0	2.08	6.33	6.0	0.58	**6.33**	6.0	0.58
12	5.33	6.0	2.08	6.33	6.0	0.58	5.67	6.0	1.53
13	4.67	5.0	1.53	4.33	4.0	1.53	5.0	5.0	1.0
14	**6.67**	**7.0**	0.58	**6.33**	**7.0**	1.15	-	-	-
All	4.84	5.0	1.74	5.0	5.0	1.81	4.71	5.0	2.06

Table 6.6: Question 1.1: Measuring the performance of reports in explaining the anomaly on a 7-point rating scale. Each case has three evaluations, they were performed by 14 out of 18 participants from the expert pool. The lowest case average for each presentation version is colored purple and the highest is colored blue.

This leads us to evaluation question 1.2, which again measures performance. To get a better comprehension of which presentation provides the most understanding, we have now asked the experts to rank the presentation versions according to their performance. These results can be found in Table 6.7. As for question 1.3 before, a descriptive analysis can be applied to these results. We assign 3 points for ranking in the first place, 2 points for second place and 1 point for third place. This shows us that *RailText* ranks highest in total.

	First place	Second place	Third place	n	Mean rank	Total Value	Ranking
Text	12 (26.7%)	14 (31.1%)	19 (45.2%)	45	2.16	83	2
TimeText	17 (37.8%)	8 (17.8%)	20 (47.6%)	45	2.07	87	3
RailText	16 (35.5%)	23 (51.1%)	3 (7.2%)	42	1.69	97	1
n	45	45	42				

Table 6.7: Question 1.2: Ranking question, measuring performance of the reports in helping the experts understand the anomaly. Total value is calculated with descriptive analysis of the ranking data. The n represent the number of evaluations.

A statistical difference between the presentations is confirmed by running a non-parametric Friedman test for all evaluation results that included all presentation versions. They are 42 out of 45 evaluations. The test resulted in a significant difference (Chi-square = 6.33,p = 0.04) (Table 6.8a). Furthermore, each presentation pair is tested with a pair-wise Conover test, which established a statistical difference between *RailText* in comparison to *Text* (p = 0.016) where *RailText* provides the better understanding of the two presentations. The other two pairs do not result in a significant difference as can be seen in Table 6.8b. No significant difference is found in the behaviour of *Text* and *TimeText* when all evaluation results are included (Cochrans's Q =0.2,p = 0.655).

Test statistic	Value
N	42
DF	2
chi-square	6.333
p	0.0402

(a) Friedman's test for cases that had all presentation version available

Pair being tested	p-value
Text - TimeText	0.5766
Text - RailText	0.0157
TimeText - RailText	0.0602

(b) Conover post-hoc test for all presentation pairs

Table 6.8: Question 1.2: Statistical tests

The same analysis format as has been discussed for the previous results is applied for the next results, where we measure the ability of the reports to represent the system. A descriptive analysis is first performed on the results. Table 6.9 contains the results. We see that the *RailText* presentation ranks highest in that analysis, but the other two presentations get the same total value. A non-parametric Friedman's test is done for all evaluations where all presentations were ranked. The test results in a significant difference between the distribution of the groups (Chi-square = 11,p = 0.003). We follow up with a pair-wise Conover test which confirms a statistical difference for *RailText* in comparison to the other two presentations (p = 0.0029, p = 0.0034), where *RailText* provides a significantly better understanding. Lastly, a pair-wise Cochran Q comparison of *Text* and *TimeText* where all evaluation results are included concludes an insignificant difference between the two (Cochran's Q = 0.36, p = 0.547).

	First place	Second place	Third place	n	Mean rank	Total Value	Ranking
Text	11(24.4%)	16 (35.5%)	18 (42.9%)	45	2.16	83	2-3
TimeText	13 (28.9%)	12 (26.7%)	20 (47.6%)	45	2.16	83	2-3
RailText	21 (46.7%)	17 (37.8%)	4 (9.5%)	42	1.59	101	1
n	45	45	42				

Table 6.9: Question 1.4: Ranking question measuring how well the reports represent the TCS. Cumulative results from three evaluation per case performed by 14 experts. Total value is calculated with descriptive analysis of the ranking data. The n represent the number of evaluations.

Test statistic	Value
N	42
DF	2
chi-square	11.0
p	0.0032

(a) Friedman's test for cases that had all presentation version available

Pair being tested	p-value
Text - TimeText	0.9539
Text - RailText	0.0029
TimeText - RailText	0.0034

(b) Conover post-hoc test for all presentation pairs

Table 6.10: Question 1.4: Statistical tests

Summary H2 To sum up our findings regarding Hypothesis 2, We have seen that a report containing graphical components does not always provide a better understanding of an anomaly. Therefore, Hypothesis 2: *"Report versions containing graphical components provide a better understanding than pure textual ones"* is not supported by our experiments. The results that give us a significant difference between presentations are all in favor of *RailText*. This tells us that *RailText* generally does provide a better understanding than pure textual reports.

H3: The same report presentation is not best suited for all cases
To answer this hypothesis we will base our findings on all the evaluation questions that have been covered up to now. The result for questions 1.1 and 1.5 that measure this hypothesis can be found in Tables 6.5 and 6.6 above. Detailed answers to evaluation questions 1.2, 1.3 and 1.4 can be found in Tables A.1 - A.3 in appendix A. By examining the tables mentioned, we can see some diversity in evaluations depending on cases and that there is a difference in the best presentation for each of them.

To get a clearer more compact overview, Table 6.11 has been constructed, containing information about which presentation version got the best results for each case and each evaluation question. If two presentations got an equally good outcome that is better than the third presentation, they are both listed. Each presentation version has been assigned a color to make the table content clearer.

Case	Q1.1	Q1.2	Q1.3	Q1.4	Q1.5
0	Text, RailText	Text	Text	Text	Text, RailText
1	TimeText	TimeText	TimeText	TimeText	TimeText , RailText
2	RailText	TimeText	RailText	TimeText	RailText
3	All Even	RailText	RailText	RailText	Text
4	TimeText	TimeText	TimeText	TimeText	TimeText
5	All Even	All Even	All Even	All Even	All Even
6	Text, RailText	RailText	RailText	RailText	Text, RailText
7	TimeText , RailText	RailText	RailText	RailText	TimeText , RailText
8	RailText	RailText	RailText	RailText	RailText
9	All Even	All Even	All Even	RailText	RailText
10	RailText	Text	Text	RailText	Text, RailText
11	TimeText , RailText	All Even	Text	RailText	TimeText
12	TimeText	All Even	All Even	All Even	Text
13	RailText	RailText	All Even	RailText	RailText
14	Text	Text	Text	Text	Text

Table 6.11: The highest performing presentation versions for all experimental cases for each evaluation question. The different presentations are highlighted with different shades of blue for visual aid.

Presentation Type	*Case*	*Anomaly Type*	*Num Element Requests*	*Num Route Requests*	*ARC Rejected*	*BevNL Rejected*	*Unexpected Route state*	*Component Crash*	*Element overlap*	*Anomalous IDCE*	*NPV overlap*
Text	**0**	-	0	2	false	false	false	false	false	false	false
	10	2	0	3	false	true	true	false	true	false	false
	14	4	3	0	false	false	false	false	false	true	false
TimeText	**1**	1	0	2	false	true	true	true	false	false	false
	2	1	0	2	false	true	true	true	false	false	false
	4	1	0	2	false	false	true	true	true	true	false
RailText	**3**	1	0	19	false	false	true	true	true	true	true
	6	2	0	2	false	true	true	false	false	false	false
	7	2	2	0	true	true	false	false	false	false	false
	8	2	2	0	false	true	false	false	true	false	false
	9	2	2	0	false	true	false	false	true	false	true
	13	3	0	1	false	true	true	false	false	true	false
Indecisive	**5**	2	0	4	true	false	true	false	true	false	true
	11	3	0	3	false	false	true	false	false	false	false
	12	3	0	4	false	true	true	false	false	false	false

Table 6.12: Experimental cases in phase 1 grouped according to presentation versions that give highest information quality. We highlight values where each property is non-zero or true.

We can see that the best performing presentations are different depending on cases. Let us investigate the ranking questions, 1.2, 1.3 and 1.4 specially and define the presentation format that performed best in those questions as the solution for the case. For cases 0, 10 and 14 *Text* is overall preferred,

TimeText seems best suited for Cases 1, 2 and 4 and *RailText* for cases 3, 6, 7, 8, 9 and 13. For the rest of the cases, it is unclear what presentations performs best. In Table 6.12 we have grouped the cases and their properties according to what presentation versions gains the highest information quality. There is no clear trend between the other properties and the most suitable presentation. This is further analyzed by calculating the Pearson correlation coefficient between the best performing presentation version and the properties of the cases. The only two significant correlation coefficients we see is a correlation of 0.82 between *TimeText* and 'Component Crash' along with *TimeText* and 'Unexpected route state' having a correlation of 0.43. All other combinations show less correlation than 0.4, either both positive or negative. Furthermore we calculated the correlation between the anomaly types and the best performing presentations, There we identify a 0.69 correlation between Anomaly type 1 and *TimeText*. Furthermore, *Text* and anomaly type 4 have a correlation of 0.53. The rest is less significant.

Summary H3 With the insights gained in this section, it is clear that a single presentation format will not suffice to provide the highest perceived information quality for any case. This provides further motivation for further hypotheses regarding the incorporation of Case-Based Reasoning to pick the most suitable presentation for each case. Hypothesis 3: *"The same report presentation is not best suited for all cases"* is therefore supported and we conclude that the same presentation is not best suited for all cases.

H4: The perceived preference of the experts is associated with the experts' work experience and their familiarity with the log output of the train control system.
Now we will investigate the connection between the reported preferences of the experts and two of their descriptive user properties, work experience with ASTRIS and log familiarity. The distribution of the cases between experts is good to keep in mind, as was discussed in Section 6.2.1. The general rule is that each expert performed 3 or 4 evaluations.

The results for questions 1.3 which measures the preference of the reports and question 1.5 which measures readability are used to answer this hypothesis. We have reorganized the results for the relevant evaluation questions by grouping according to familiarity, work experience and expert clusters. The reorganizations can be found in Tables 6.13, 6.15 and 6.17 for question 1.3. For question 1.5 the reorganized results can be found Tables 6.14, 6.16 and 6.18.

Work Experience The work experience of the experts with ASTRIS is first considered. In Table 6.13 the results for questions 1.3, measuring preference of the report presentations, have been sorted into four groups according to the work experience of the experts that gave the evaluations. We apply a descriptive analysis as was defined in Section 6.2.7, similarly to what we have done in previous ranking questions. By looking at the total values and ranking in the table we see that there is some difference in the ranking outcome depending on the work experience group. But there does not seem to be a general rule that explains that behavior in relation to the work experience of the experts.

The calculated total values indicate in most of the groups that there is not a large variation on the preferences for the different presentations. We have run Friedman's test for each of the work experience groups individually, that resulted in an insignificant difference in behaviour of the presentations for all groups (p-values for the groups in the order of increasing work experience, 0.34, 0.053, 0.685, 0.14). So the null-hypothesis, which says that there is no difference between the distributions of the presentations within each group, is not rejected for any of the groups. Which means there is no significant difference for the presentation versions for any work experience group.

In Table 6.14 we again have results grouped by work experience but now for question 1.5 measuring readability on a 7-point rating scale. The highest value for each expert group is highlighted in the table. For all groups, except those that have a work experience of 2 to 3 years, *Text* presentation provides the most readable report. In the final group, *TimeText* is evaluated as the most readable. These results are somewhat inconclusive since the distinction between presentation average values is most often smaller than the standard deviation of the values. Moreover, there is a big difference in the sizes of the groups. Additionally, the difference between the highest and second-highest values are quite small for all groups. ANOVA tests were run for each of the evaluation groups, where no test resulted in a significant variance between presentation versions within a group; 1-2 years: ($F(2,17) = 0.26$, $p =$

Work experience	Num. experts	Num. evaluations	Presentation	First place	Second place	Third place	Total Value	Ranking
1 - 2 years	2	7	Text	1 (14.3%)	4 (57.1%)	2 (33.3%)	13	2-3
			TimeText	3 (42.9%)	- (0.0%)	4 (66.7%)	13	2-3
			RailText	3 (42.9%)	3 (42.9%)	- (0.0%)	15	1
2 - 3 years	2	7	Text	1 (14.3%)	- (0.0%)	6 (85.7%)	9	3
			TimeText	3 (42.9%)	4 (57.1%)	- (0.0%)	17	1
			RailText	3 (42.9%)	3 (42.9%)	1 (14.3%)	16	2
3 - 4 years	3	8	Text	3 (37.5%)	2 (25.0%)	3 (42.9%)	16	1-2
			TimeText	3 (37.5%)	1 (12.5%)	4 (57.1%)	15	3
			RailText	2 (25.0%)	5 (62.5%)	- (0.0%)	16	1-2
More than 4 years	7	23	Text	9 (39.1%)	7 (30.4%)	7 (31.8%)	48	2
			TimeText	6 (26.1%)	4 (17.4%)	13 (59.1%)	39	3
			RailText	8 (34.8%)	12 (52.2%)	2 (9.1%)	50	1

Table 6.13: Question 1.3: Ranking of preference of presentations, grouped by work experience. The n represents the number of evaluations that fall within the category.

0.772), 2-3 years: (F(2,18) = 1.61, p = 0.228), 3-4 years: (F(2,20) = 0.758, p = 0.48) and more than 4 years: (F(2,65) = 1.87, p = 0.163).

			Text			TimeText			RailText		
Work experience	Num. experts	Num. evaluations	Avg	Med	Std	Avg	Med	Std	Avg	Med	Std
1 - 2 years	2	7	**4.71**	5.0	0.49	4.43	4.0	0.98	4.0	5.0	1.91
2 - 3 years	2	7	4.0	4.0	1.15	**5.14**	6.0	1.21	4.86	5.0	1.35
3 - 4 years	3	8	**5.25**	5.0	0.89	4.75	4.0	1.04	4.62	5.0	2.07
> 4 years	7	23	**4.61**	5.0	1.67	3.83	4.0	1.77	4.57	5.0	2.06

Table 6.14: Question 1.5: Readability evaluated on a 7-point rating scale. Grouped by work experience. The highest value for each work experience group is highlighted.

Log familiarity Next, we see how the log file familiarity of the experts influences their preferences. The familiarity is on a scale from 1 to 5, where 1 represents *Not very familiar* and 5 *very familiar*. For these results, it is important to draw note that the size of the groups according to data familiarity is very uneven. Where the vast majority of expert has a high level of data familiarity.

First, the ranking according to the preference of the reports is studied from results in Table 6.15. Familiarity group 1 has a clear difference between presentations (Friedman's chi-square = 6, $p < 0.0001$), where *RailText* is most preferred. Since this group only holds evaluations from one expert we are unable to say that that is a general rule for experts with low familiarity.

Familiarity group 2 consists of only one expert as well. There we can see that Text is always in the lowest ranking but the presentations containing a graphical component are alternately in the first and second place. By running a non-parametric Friedman's test it is confirmed that a significant difference is in the behavior of some presentation groups (Friedman's chi-square = 4.67, p = 0.049). Furthermore pairwise comparisons show that the difference lies in *Text* versus *TimeText* (p = 0.0241) and *Text* vs *RailText* (p = 0.0474).

In familiarity groups 4 and 5 we have more experts and evaluations. That leads to a higher variance of the rankings per each group so the total values calculated by the statistical analysis described in Section 6.2.7 are somewhat close to each other. Familiarity group 4 has the highest preference for Text presentation, but the difference between groups is not significant (Friedman's chi-square = 2.18, p = 0.35). A similar outcome can be seen for familiarity group 5. *RailText* has the highest overall value according to the ranking, but the difference is again insignificant (Friedman's chi-square = 3.12, p = 0.213).

Data Familiarity	Num. experts	Num. evaluations	Presentation	First place	Second place	Third place	Total Value	Ranking
1	1	3	Text	- (0.0%)	- (0.0%)	3 (100.0%)	3	3
			TimeText	- (0.0%)	3 (100.0%)	- (0.0%)	6	2
			RailText	3 (100.0%)	- (0.0%)	- (0.0%)	9	1
2	1	3	Text	- (0.0%)	- (0.0%)	3 (100.0%)	3	3
			TimeText	2 (66.7%)	1 (33.3%)	- (0.0%)	8	1
			RailText	1 (33.3%)	2 (66.7%)	- (0.0%)	7	2
4	4	13	Text	5 (38.5%)	6 (46.2%)	2 (18.2%)	29	1
			TimeText	4 (30.8%)	2 (15.4%)	7 (63.6%)	23	3
			RailText	4 (30.8%)	5 (38.5%)	2 (18.2%)	24	2
5	8	26	Text	9 (34.6%)	7 (26.9%)	10 (40.0%)	51	2
			TimeText	9 (34.6%)	3 (11.5%)	14 (56.0%)	47	3
			RailText	8 (30.8%)	16 (61.5%)	1 (4.0%)	57	1

Table 6.15: Question 1.3: Ranking of presentations - preference, grouped by data familiarity

We also look into the readability in terms of the familiarity values of the experts. By observing the results found in Table 6.16 we see the highest value for each group highlighted. It seems that experts that have a higher familiarity prefer *Text* presentation and the ones with low familiarly prefer the graphical presentations. These findings are tested with one-way ANOVA for each expert group. That results in a significant difference between presentations for familiarity groups 1($F(2,6) = 12$, $p = 0.008$) and group 2 ($F(2,6) = 7$, $p = 0.027$). In both cases the evaluation observations are however so few that the significance of the ANOVA is no fully conclusive. Within the higher familiarity groups we do not detect a significant difference between presentations, familiarity group 4 ($F(2,36) = 0.61$, $p = 0.546$) and group 2($F(2,75) = 1.31$, $p = 0.27$).

			Text			TimeText			RailText		
Data Familiarity	Num. experts	Num. evaluations	Avg	Med	Std	Avg	Med	Std	Avg	Med	Std
1	1	3	3.67	4.0	0.58	5.67	6.0	0.58	5.67	6.0	0.58
2	1	3	5.0	5.0	0.0	5.33	5.0	0.58	6.0	6.0	0.0
4	4	13	4.15	5.0	1.95	3.54	4.0	1.61	3.31	5.0	2.39
5	8	26	4.96	5.0	1.04	4.38	4.5	1.47	4.85	5.0	1.52

Table 6.16: Question 1.5: Readability evaluated on a 7-point rating scale. Grouped by familiarity. The highest value for each familiarity group is highlighted.

Expert clusters We also look into the distribution of the expert properties in terms of the expert groups identified in Section 4.1.1. Similarly as was done in the initial study, we perform a k-means clustering algorithm to identify the most distinct clusters in the distribution of the experts according to their work experience and log file familiarity. The three clusters identified are similar to the ones detected in the initial study, which can be seen in Figure 4.3. The group represent the same trends as were identified as the main trends in the expert groups previously found. That is:

- Group 1 consists of experts that have a high familiarity with the log files and less work experience .
- Group 2 consists of experts that have both high familiarity and great work experience.
- Group 3 consists of experts that have low familiarity with the log files.

After having grouped the evaluators into these expert groups, we again have groups that hold a very different amount of experts. The majority, or 9 out of 14 of the expert belong to Group 2 that represent experts that have both high log file familiarity and high work experience. Three experts belong to group 1 and two belong to group 3.

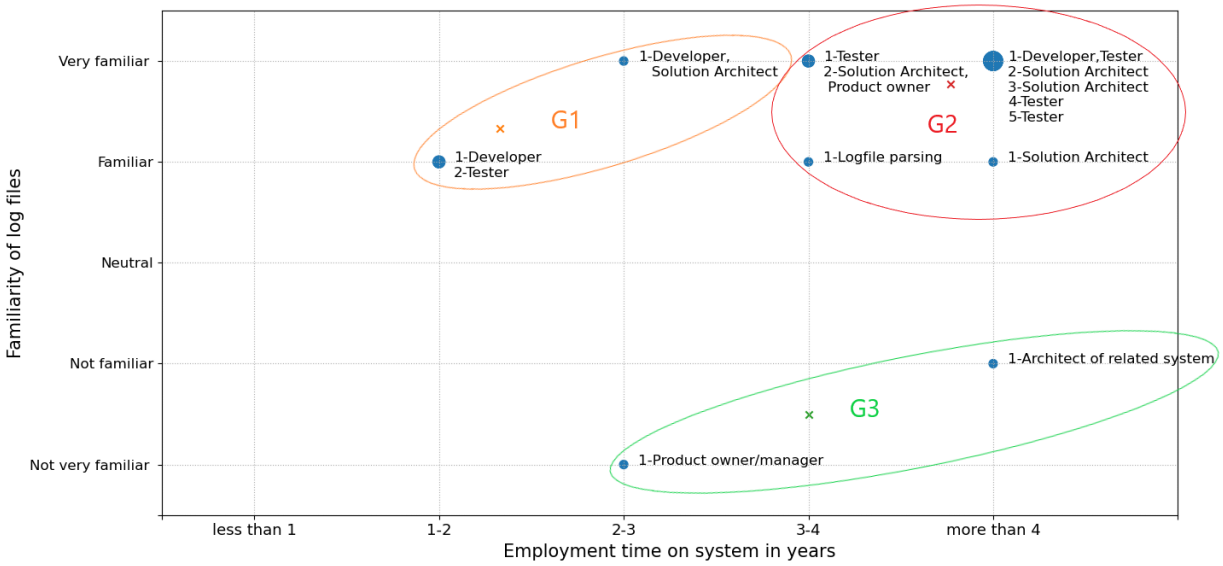


Figure 6.5: Combined overview of worker properties

First, we look into the ranking according to preference which can be seen in Table 6.17. Group 1 has no statistical difference on preference depending on the presentation version (Friedman's chi-square = 1.4, p =0.5204). Group 2 prefers *RailText* and *Text* almost equally. We detect a slight significant difference between the presentation versions within group 2 when running the Friedman's Chi-square test(Friedman's chi-square = 6.1, p < 0.0447). By running a pairwise comparison we see the significance lies in the difference between *TimeText* and *RailText*(p = 0.0174). Group 3 seems to have a clear preference for presentation versions containing graphical components over the purely textual presentation. A significant difference is confirmed (Friedman's chi-square = 9.33, p < 0.001). By running a pairwise comparison we see that there is a significant difference in preference for *Text* in comparison to the other two presentations, *Text* vs *TimeText*(p = 0.0012) and *Text* vs *RailText* (p<0.001). *Text* is the less preferred in both cases.

Data Familiarity	Num. experts	Num. evaluations	Presentation	First place	Second place	Third place	Total Value	Ranking
Group 1	3	11	Text	4 (36.4%)	4 (36.4%)	3 (27.3%)	23	1-2
			TimeText	4 (36.4%)	1 (9.1%)	6 (54.5%)	20	3
			RailText	3 (27.3%)	6 (54.5%)	2 (18.2%)	23	1-2
Group 2	9	28	Text	10 (35.7%)	9 (32.1%)	9 (36.0%)	57	2
			TimeText	9 (32.1%)	4 (14.3%)	15 (60.0%)	50	3
			RailText	9 (32.1%)	15 (53.6%)	1 (4.0%)	58	1
Group 3	2	6	Text	- (0.0%)	- (0.0%)	6 (100.0%)	6	3
			TimeText	2 (33.3%)	4 (66.7%)	- (0.0%)	14	2
			RailText	4 (66.7%)	2 (33.3%)	- (0.0%)	16	1

Table 6.17: Question 1.3: Ranking of presentations - preference, grouped by expert groups

In Table 6.18 we see the evaluation of the reports in terms of readability. *RailText* yields the highest score according to Groups 1 and 3 but expert group 2 thinks Text presentation is most readable. Those findings do not reflect the whole story since the standard deviations of the averages are all relatively high, except for expert group 3. We therefore run a one-way ANOVA on the results within each group. That results in a insignificant difference between the presentation version for group 1 (F(2,30) = 0.45, p = 0.64) and group 2 (F(2,81) = 1.48, p = 0.23). However, for group 3 we detect a significant difference between the performance of presentations (F(2,15) = 9.85, p = 0.0019). The difference lies between

Text and the other two versions. *Text* vs *TimeText* (F(1,10) = 8.44, p = 0.0157) and *Text* vs *RailText* (F(1,10) = 16.2, p = 0.0024). Text provides less readability in both cases.

			Text			TimeText			RailText		
Work experience	Num. experts	Num. evaluations	Avg	Med	Std	Avg	Med	Std	Avg	Med	Std
Group 1	3	11	3.82	5.0	1.89	3.73	4.0	2.0	**4.09**	5.0	2.12
Group 2	9	28	**5.04**	5.0	1.07	4.25	4.0	1.35	4.43	5.0	1.93
Group 3	2	6	4.33	4.5	0.82	5.5	5.5	0.55	**5.83**	6.0	0.41

Table 6.18: Question 1.5: Readability evaluated on a 7-point rating scale. Grouped by expert groups. The highest value for each expert group is highlighted.

Summary H4 We detected no significant difference in the distribution of presentation versions when evaluators are grouped according to work experience. When grouping evaluators according to log familiarity, those that have a low familiarity seem to rather prefer graphical presentations, however that is an under-represented group so the significance is inconclusive. Finally, when clustering the evaluators according to both expert properties, we again see that the evaluator group consisting of experts with lower familiarity have a preference for presentation versions containing graphical components. We also see that in terms of preference that evaluators with high familiarity and experience do not prefer *TimeText*.

We do not detect any association between work experience and perceived preference of experts. Therefore, Hypothesis 4: *"The perceived preference of the experts is associated with the experts' work experience and their familiarity with the log output of the train control system."*, is not supported by our experiments. However, we have detected an association between familiarity of log files and perceived preference. Still, this is an under-represented group and it would thus be better to have more evaluators with low familiarity for further studies.

Preference of reports over raw log files

This does not test any hypothesis but measures the usefulness of the reports and is evaluated with question 1.6. We see that 25 times the expert evaluators said that would prefer to use the reports over raw log files. Only 6 answered this negatively and the rest was not sure. However a much larger part of the experts answered positively, then negatively so we see that there is a general preference for using the generated reports to understand anomalous incidents. The significance of these results were tested with a Cochran's Q test which resulted in a statistical difference of the results (Cochran's Q=11.2, p = 0.0037). Where the main difference in observations lies in between the positive and negative options(Cochran's Q=10.2, p =0.0015). We therefor conclude that the reports are in general useful for the experts.

Case	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**Sum**
Yes	1	2	3	-	2	-	1	2	2	2	2	2	2	1	3	25
Maybe	-	1	-	2	-	3	-	1	1	1	1	1	-	2	-	13
No	2	-	-	1	1	-	2	-	-	-	-	-	1	-	-	6

Table 6.19: Question 1.6: Measuring the usefulness of the reports over raw log files. Each case is evaluated three times by different experts. The numbers represent a count for each option being chosen.

Other Insights

In 57,8 % of the evaluation cases, the experts made use of this optional question as a platform to provide some kind of feedback. This resulted in 26 answers. The open question provided various insightful feedback. Mostly, the reasons for lower evaluation scores were due to additional requirements listed for the reports. Some of these 26 answers gave examples of possible functionality that could be incorporated in the system while others suggested the inclusion of information that is not available to

us. Many of the answers are a good indication of requirements that could be added if the system will be further enhanced and even more of the log file types available included in future research on the project. We will perform an analysis of the answers to further gain insights from them. We start by extracting codes from each answer that identify the core concepts of the answer. The coding is performed on sentence-level so each answer can be represented with multiple codes if there are multiple sentences in an answer and they cover various topics. The coding procedure results in 23 different codes which can be seen in Table 6.20. Each code occurs between 1 and 8 times in the answers. In total, we get a count of 44 code occurrences, where they range from 1 to 4 assigned to each answer.

Code	Count	Category
• Improvement of timelines	8	Improvement ideas
• Different annotation for railways	3	
• Improvements on overview in beginning of reports	2	
• Interactive system where users can pick what information should be included in the reports.	2	Ideas for further research
• Combine report versions	1	
• Use reports and raw log files	1	
• Data tables should not be included with graphical components	1	
• Use less sentences and more bullet points	1	
• Too much information presented	4	Complaints
• Relationship between log entries should be better explained	3	
• Important field for an anomaly is missing	2	
• Color code used in other system disturbs interpretation of railway overview	1	
• Text layout can be confusing	1	
• Larger time frame needed	1	
• Expected behaviour should be clearer	1	
• Shared elements should be more clearly noted	1	
• Explaining why reports are helpful	4	Praise
• Insights in when each presentation is suitable are useful.	3	Insights to expert perception
• Difficult to read reports when they are information dense	2	
• Data missing that is not available in our log entries	1	Irrelevant
• Comment on changed functionality in different TCS versions	1	

Table 6.20: Overview of codes extracted from open question accompanied with their count and category.

We have furthermore identified some themes in the codes, which allowed us to group the found remarks in six main categories. An answer from one expert can belong to one or more categories, depending on how the sentences in the response were coded. We will now go through each of the categories and inspect what the codes within that category represent.

1. **Improvement ideas** The sentences that have been grouped to this category are all providing ideas and improvements that could relatively easily be applied to the current report generation system. Therefore they are counted as improvement ideas instead of future work.

   8 of them concern how the graphical *timelines* can be improved:

   - Some experts pointed out small aspects of the timelines that could be easily adjusted to meet their requirements, that were adjusted for the next evaluation phase. Those were things such as the lack of a guide to understand the different colors of element timelines, which indicate different log entry types and that the placement of some timelines was to close to the text prior, which cause an incidental overlap.
   - In smaller incidents, it can occur that elements only have one entry. When that happens the timeline should not be included.
   - The different scales of timelines that are applied to ensure that annotations have as much space as possible can cause matching between timelines to be difficult.
   - Text in timeline annotation should be larger.

- One expert suggested the timelines would be combined for elements instead of presenting one timeline for each element

Three answers concerned possible improvements for *railway* components:

- Use different colors for the annotations since the color that represents switches seems to indicate an anomaly.
- Highlight better what data has changed between different time frames of the railway overview.

Lastly, two answers concerned improvements for the overview at the beginning of the reports:

- One expert proposed that all anomalies should be reported in an overview at the beginning of reports. This is currently done separately for each request, but might be informative to include an additional overview showing all anomalies.
- Another expert wanted to add the request identification to the request overview in the first component of all reports.

2. **Ides for further research**

   Some answers suggest a different design focus for the system. For example with a combination of the available modalities. One answer specified that the expert would like to use such reports along with raw log files, instead of having to pick one. Another one specified that he would prefer a report presentation containing both railway overviews and timelines. An interactive system was also suggested by two experts. In such a system, the experts could pick what information they want to be portrayed in the reports and how detailed the information should be.

   Other answers concerned more generic aspects of the report design. One expert said to have a preference for more text on a bullet format instead of sentence format, which could be interesting to compare with each other. Lastly it was suggested to remove data tables from the *TimeText* and *RailText* presentation versions. That is now considered part of the textual core of the reports and therefore part of all presentation versions

3. **Complaints**

   This category contains answers that point out aspects of the reports that the experts did not like or thought that certain aspects were lacking.

   - Three answers mention that too much information is included in the reports. So content selection should be reconsidered in future work. This concerned i.e. information that explains the configuration of railway routes and reporting of element states that are likely to cause anomalies.
   - Important data field for some anomaly types are missing according to two experts. This was for case 5 where the timeout of a BevNL command should be included and case 14 where a field “Actuel Gebruik” is said to be missing in all report presentations.
   - Relationship between different log entries should be better explained.
   - Some colors used to annotate railway overviews are similar to colors that have another meaning in other software used by a part of the expert. That caused an expert to draw incorrect meaning from the railway overview.
   - The text layout in the reports was confusing to one expert.
   - Time frame should be larger for some cases. explained.
   - One expert thinks the expected behavior of the system should be better explained in the reports. However that was not the goal of the system, this could be interesting to add in future work but will need more extensive domain knowledge.
   - The overlap of elements between routes is not displayed well enough.

4. **Praise** This category contains answers where certain aspects of the reports were praised. Some of the praised aspects are listed up here beneath:

- The report with the railway is really helpful since the railway overview adds environmental context.
- The reports indicate possible problems. It helps with the first steps of a problem analysis by gathering information and making a timeline. The report partially automates the work, saving valuable time.
- The data tables that can be found in all reports are considered useful.
- Lastly the reports, in general, are said to be helpful for the experts if they would be generated.

5. **Insights to the expert perception of reports** Answers where experts provided deeper insights to explain their choices fall within this category. We will now go through some insights gained from their answers.

   - Pure textual presentation is better than the others when the amount of messages conveyed in a report is small. It also allows you to get a more compact overview of the cases.
   - Railway overviews are informative when an environmental context is not needed.
   - The timelines are a bit overwhelming to some experts, which causes them to be less informative.
   - Lastly the reports are twice said to be less helpful in information-dense cases when many requests are being handled.

6. **Irrelevant**

   2 answers belong to this category. This category contains answers that pointed out aspects that were not within the scope of this thesis project.

   Answers that fall within this category pointed out aspects of the reports that are out of context for the project. They ask for information that is not to be found within the log files that are in the scope of the project or refer to an older version of some layer of the TCS.

   The log files that are indicated would be a good extension to the system in future work.

This overview shows us that the focus is not the same for all experts and the ideas to improve the system are very diverse. Therefore the interactive suggestions might be most interesting to put in the focus of the next phase of research concerning this system, and hopefully being able to offer all of the other functionality suggested in the interactive selection for data and presentation options.

6.3. Case-Based Reasoning application

To be able to use a CBR approach, we need a case base containing solved problems, where the properties of the problems and their solutions are stored. CBR systems learn from previously solved problems, that are stored in a case base, and can apply that knowledge to solve new unseen problems.

Since our domain of interest for this project does not have any predefined solutions for this, a case base has to be built. The results from this evaluation phase are used to form our case base, where the best presentation per each case is found from the evaluation answers. The case base will be further used to predict solutions for new experimental cases to gain insights to answer our research questions concerning the usage and performance of our CBR system.

6.3.1. Generation of case base

We base our findings for a solution for each case on a combination of evaluation questions asked in phase 1, that answer various aspects of which presentation version is most suitable. We first look at the three ranking questions asked, since the experts have directly ranked the presentation versions there according to various metrics. If two or three presentation versions are equal in ranking, we look at the values given for the independent evaluation of the presentations in questions 1.1 and 1.5. This was needed for determining a solution for cases 5,11 and 12. There we determine which of the equally ranked presentations has the highest rating. If they are still equal, then no solution is defined for that case. This was the result for cases 5 and 12. Otherwise the found solution is assigned to the case as a solution in the case base. Table 6.21 shows the presentation versions that got ranked highest in each category for each case.

Case	Highest ranked in helping understanding the anomaly	Highest ranked in preference to use for this anomaly	Highest ranked in representing the system	Solution
0	Text	Text	Text	Text
1	TimeText	TimeText	TimeText	TimeText
2	TimeText	RailText	TimeText	TimeText
3	RailText	RailText	RailText	RailText
4	TimeText	TimeText	TimeText	TimeText
5	-	-	-	No Solution
6	RailText	RailText	RailText	RailText
7	RailText	RailText	RailText	RailText
8	RailText	RailText	RailText	RailText
9	-	-	RailText	RailText
10	Text	Text	RailText	Text
11	-	Text	RailText	RailText
12	-	-	-	No Solution
13	RailText	-	RailText	RailText
14	Text	Text	Text	Text

Table 6.21: Presentation versions that influence the final choice of a solution for the cases. If no presentation had a majority vote by its evaluators it is marked with '-' and not considered in the final decision for a presentation solution for the case.

Two cases, number 5 and 12 did not yield a solution, since no presentation was dominant over the others. They are not inserted to the case base since they do not have a solution that the case base can learn from. The other 13 cases construct the initial cases for the case base. Each case is inserted with its properties described in Table 6.1. In our design the case base is stored in a CSV file. The CBR system therefore now knows what solution is most suitable for each combination of properties that can be found in the experimental cases for phase 1.

By extracting the most suitable solution for all the cases we see that the solution presentation varies considerably between cases as was briefly discussed concerning H4. So there is indeed a good reason for us being looking into these different modalities for the presentation versions.

6.3.2. Predicting presentations for Phase 2 cases

Now the case base that was just constructed is used to predict the suitable presentation version for 14 new cases. A Case-Based Reasoning(CBR) system is built out of 4 components, retrieve, reuse, revise and retain, as has been described earlier in our literature review and methodology Sections 2.4 and 5.7.2. We go through each of the components, for one case at a time. Since we want to be able to gain more knowledge and allow the system to learn more after each iteration. The results for each case can be found here below in Table 6.22.

First, we retrieve the case from the case base that is most similar to the new incoming case according to the Mahalanobis distance metric. The solution template that can be found in the case base for the retrieved case is then used to generate a report for the incoming case. To ensure that solution is indeed suitable, a small revision phase was performed. This is done to be able to enhance the case base by each new solution found. For that to be possible the solution needs to be approved or revised before being retained to the case base. This small evaluation phase is performed by the CGI thesis supervisor. In this revision phase, he was asked to evaluate whether the CBR system managed to find the most suitable presentation for each case.

During the process of finding the most suitable presentation for the new experimental cases, he found 4 cases that he did not think the CBR system managed to find the right solution for. Cases 15 and 16 were revised from *Railway* to *Text* simply because the nature of the incidents does not provide a railway representation. Cases 21 was revised from *Railway* to *Timelines* on the ground of the incidents not representing routes. Case 22 was revised from *Text* to *Timelines* since that was considered more suitable

After each case had been revised it was retained. If the presentation choice was changed during the revision phase, the revised presentation is retained to the case base to enhance its knowledge.

If no revision act was needed the presentation version that was retrieved from the case base will be assigned to the new case and then retained to the case base.

Case	Retrieval	Reuse	Revision	Retention (Chosen presentation)
15	RailText	RailText	Text	Text
16	RailText	RailText	Text	Text
17	RailText	RailText	-	RailText
18	RailText	RailText	-	RailText
19	RailText	RailText	-	RailText
20	RailText	RailText	-	RailText
21	RailText	RailText	TimeText	TimeText
22	Text	Text	TimeText	TimeText
23	RailText	RailText	-	RailText
24	RailText	RailText	-	RailText
25	RailText	RailText	-	RailText
26	RailText	RailText	-	RailText
27	Text	Text	-	Text
28	RailText	RailText	-	RailText

Table 6.22: The output of all phases of the CBR system for the cases that will be evaluated in Phase 2.

6.4. Phase 2

In this phase we will investigate cases number 15 to 28 which are new, unseen cases that were processed with the CBR system in Section 6.3.2 above. We will see how the domain experts evaluate the report generated using the presentation template that the CBR system chose for each case and how well that presentation manages to meet the goals of the system.

Similarly to phase 1 we will cover the dependent variables, materials, evaluators, questions, hypotheses, execution and results. The independent variables described in Section 6.1 apply to this phase as well.

6.4.1. Dependent variables

For this evaluation phase, we again need to consider some aspects to make sure the design is not biased towards any of them.

In this phase, 14 cases are being evaluated by three experts. Each expert evaluates all 14 cases. The overview of these cases and their properties can be found in Table 6.23. To avoid order bias we again apply a Latin Square design for the order of the cases presented to each evaluator. Each evaluator gets a unique arrangement of the cases, so the outcome should not be influenced by the position of the case in the evaluation order.

The dependent variables evaluated will now be introduced.

Prediction Precision First, we measure the prediction precision of the CBR system. The correctness of a prediction is measured by asking the experts whether they would rather prefer another report presentation than the CBR system predicted.

- *If you would rather prefer another report presentation for the anomaly scenario of interest, which report presentation would you prefer?*

Perceived Understanding Next, we measure the perceived understanding of the chosen report presentation. To measure this we make use of two questions

- The performance of the chosen report presentation is first measured. *To what extent do you feel this report explains the incident?*
- The readability of the chosen report presentation is next measured. *How clear in terms of readability do you think the report is?*

Usefulness Furthermore, we measure the usefulness of the chosen presentation by asking whether the experts would prefer to use it instead of raw log files. As in Phase 1, this is not included to answer our research questions, but to get an indication of whether we are meeting the goal of the project in terms of CGI.

- *Would you prefer to use any of those reports over raw log files to understand the anomaly scenario of interest?*

Meeting Requirements Lastly, we measure how well the chosen presentation manages to meet the defined requirements of the anomaly types in focus in this research. This is again not connected to a research question but to measure how well the system managed to follow its requirements.

- *To what extent does the report meet the requirements of the incident type?*

6.4.2. Materials

For each of the experimental cases being evaluated in this evaluation phase, we have generated all three presentation versions. Additionally, the CBR system has predicted the most suitable presentation for all cases. The chosen presentations can be found in the last column of Table 6.22. There we can see that 2 out of 14 (14.3%) cases resulted in a *TimeText* presentation, 3 out of 14 (21.4%) in a *Text* presentation and the rest, or 9 out of 14 (64.3%) in a *RailText* presentation.

Case	Anomaly Type	Num Element Requests	Num Route Requests	ARC Rejected	BevNL Rejected	Unexpected Route state	Component Crash	Element overlap	Anomalous IDCE	NPV overlap
15	1	0	0	false	false	false	true	false	false	false
16	1	1	0	false	false	false	true	false	false	false
17	1	0	12	false	false	true	true	true	true	false
18	2	0	2	false	true	true	false	true	false	false
19	2	0	1	true	false	true	false	false	false	false
20	2	1	1	true	false	true	false	false	false	false
21	2	1	0	true	true	false	false	false	false	false
22	2	1	0	false	false	false	false	false	true	false
23	2	0	1	false	true	true	false	false	false	false
24	3	0	1	false	true	true	false	false	false	false
25	3	0	2	false	false	true	false	false	false	false
26	3	0	2	false	false	true	false	true	false	true
27	4	3	0	true	false	false	false	true	false	false
28	4	1	0	true	false	false	false	false	false	true

Table 6.23: Overview of cases evaluated in phase 2. We group the cases by anomaly types and highlight values where each property is non-zero or true.

All experimental cases are visualized with regards to their Boolean properties in Figure 6.6. The figure has the same format as the analogous Figure 6.1. We see that case 23 and 24 have the exact same property combination, even though they do not belong to the same anomaly type. That can be explained by the overlap that is possible between anomaly types 2 and 3, since they can both contain route related anomalies, but the origin of them is what distinguishes them. This example demonstrates how important the properties are to understand what happens within an anomalous case.

6.4.3. Evaluators

Due to circumstances, many of the domain experts were extremely busy during the time that the evaluations took place. Therefore, three experts from the original expert pool were asked to participate in the final evaluation. This resulted in each expert evaluating 14 cases. The experts represent the average

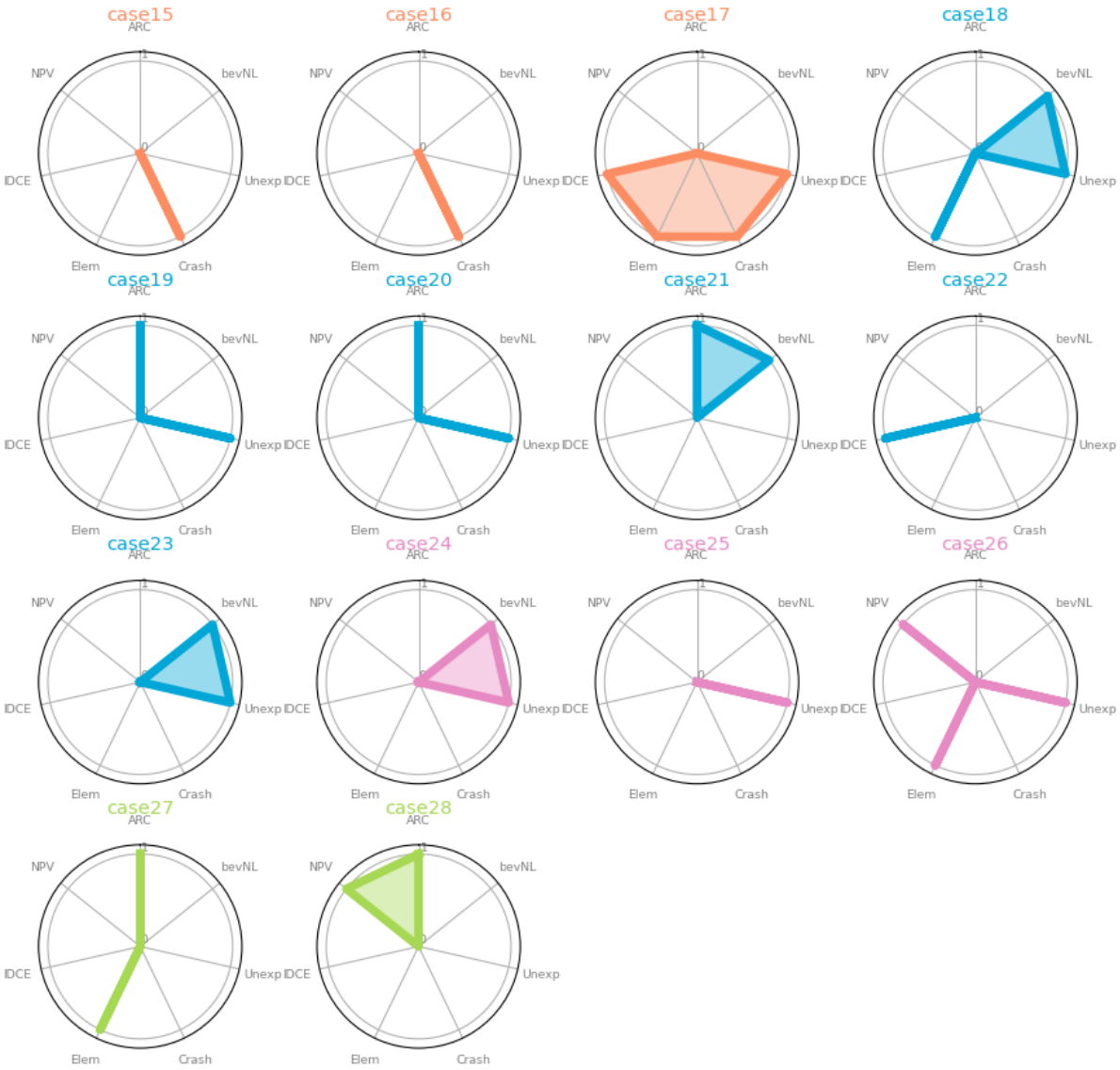


Figure 6.6: Visualisation of case properties.The different colors represent different anomaly types, anomaly type 1 is orange, 2 is blue, 3 is pink and 4 is green.

log familiarity of the expert pool, but not the average work experience. In the pool of evaluators, two have the data familiarity of 4 out of 5 and one has the maximum of 5. Two have been working on the system for 1 to 2 years, while one has a work experience of over 4 years on the system. Finally, we have two developers of the system and one tester. This is not a perfect representation of the whole expert group, but should give us some insights into the opinion of the expert pool.

6.4.4. Evaluation questions

Again we categorize the questions by whether they are extrinsic, intrinsic or neither. The study is in a within-study format again, but with a specific focus on the report presentations that were chosen as most suitable for each case by the CBR system. Here below we will refer to the presentation that was chosen as the “Chosen presentation”.

An overview Table 6.24 that shows the relationship between research questions and evaluation questions can be found in Section 6.4.5.

Extrinsic evaluation
Here we measure how well the system manages to perform its task. We mostly focus on the result of the CBR system, but by doing that we also evaluate the result of the report generation itself.

We start by evaluating the **performance** of the chosen presentation. That is the same question as Question 1.1 and has the same format and scale.

- **Question 2.1:** *"Chosen presentation: xxx. To what extent do you feel this report explains the incident?"*

Next, we wanted to check how well the report chosen by the CBR system manages **meeting the requirements** that were defined in 4.3.1. Here the requirements for the relevant anomaly type were provided for the experts to evaluate according to. The setup of a rating scale question can be seen in Figure 6.3. The expert was asked to rate the reports on a 7-point rating scale on the scale from "Very well" to "Very Poorly".

The last extrinsic task is to evaluate the **performance of CBR system**, by testing whether the CBR system manages to find the most appropriate report presentation for each case. This is an optional question where the experts get the presentation versions that were not selected by CBR choose from.

- **Question 2.3:** *"If you would rather prefer another report presentation for the anomaly scenario of interest, which report presentation would you prefer?"*

For question 2.3, we make sure to randomize the order of the other presentation versions to choose from so that not to cause order bias.

Intrinsic evaluation
The intrinsic questions asked here are the same as the intrinsic questions in phase 2, apart from the questions only apply to the chosen presentation, not all presentation versions.

We first measure the **readability** of the chosen report. This is the same question as Question 1.5 and has the same format and scale.

- **Question 2.4:** *"Chosen presentation: xxx . How clear in terms of readability do you think the report is?"*

Next, we measure the **usefulness** of the chosen report.

- **Question 2.5:** *"Would you prefer to use the report presentation given by the system over raw log files to understand the anomaly scenario of interest?"*

The options given for this question are "Yes", "No" and "Maybe".

Free Text Evaluation
Again an open question was included to give the opportunity for open feedback from the experts.

- **Question 2.6:** *"Any additional comments or remarks regarding the reports for this incident case?"*

6.4.5. Hypotheses
In this evaluation phase we test a few hypotheses.

Hypothesis 5 (H5): *The CBR system can find the report presentation that is most preferred by the experts for the majority of cases.*

H5 concerns the performance of the CBR system and is tested by question 2.3 using extrinsic evaluation.

Hypothesis 6 (H6): *The report presentation chosen by the CBR system helps the experts understand the anomalous incidents.*

H6 concerns the performance, the readability and the usefulness of the chosen report. This is tested with evaluation questions 2.1, 2.4 and 2.5. Questions 2.1 and 2.4 are rated on a 7-point scale. The higher the rating is, the more helpful the report is considered. Question 2.5 on the other hand only

has three options “Yes”, “No” and “Maybe”. There “Yes” is considered helpful, “No” is not helpful and “Maybe” is neutral.

Table 6.24 provides us with an overview of the connection between all research questions, hypotheses and research questions and is meant to help us navigate through the current chapter.

Research question	Hypotheses tested	Criteria measured
sub-RQ3 How is Case-Based Reasoning, as an adaptive template selection method for anomalous incidents, where incidents are represented with variables, perceived by experts?	**H5** The CBR system can find the report presentation that is most preferred by the experts for the majority of cases.	Performance of CBR
	H6 The report presentation chosen by the CBR system helps the experts understand the anomalous incidents.	Performance, Readability & Usefulness

Table 6.24: Overview of the relationship between research questions, hypotheses and evaluation questions relevant to Phase 2

6.4.6. Execution

This evaluation phase was carried out in the same format as phase 1. The questionnaire consists of the same components as where described in Section 6.2.6, namely introduction, Consent, User Model and Report Evaluation. The description of the first three components is the same here as in phase 1 but the report evaluation has a different format.

Report Evaluation Lastly, the reports are evaluated. For each case, the anomaly definition is given along with the report version selected by the CBR system which is presented as the “Chosen Presentation”. The chosen presentation along with the other report presentations are provided through an internet link. Here we focus on evaluating the outcome of the CBR system and see whether the choice of report presentation was correctly predicted by the CBR system. This is evaluated through the questions explained above in Section 6.4.4.

After the execution of this evaluation phase, it was observed that two of the anomaly case descriptions did not completely match the cases created. For case 17 a wrong ASTRIS software component was said to have crashed, it was the IDCR component that crashed but by a mistake the route component was indicated as crashing in the case description. In case 19, a wrong type of command was listed as being processed in the case. This might influence our results as the reports do not explain the exact behaviour listed in the descriptions. We will however include these evaluation results for these cases in our result analysis since they still include some insights, but keep these description faults in mind while we analyze. Furthermore, in a real world situation this description would not be available. The description would be detected from the reports. So it will be valuable to see if the real anomalous behaviour can be detected from the reports when the description is not correct.

6.4.7. Results

H5: The CBR system can find the report presentation that is most preferred by the experts in the majority of cases

In question 2.3, the experts were given what the chosen report presentation by the CBR system is, for each case being evaluated. They were asked if they would prefer either of the other two over the chosen presentation instead. In Table 6.25 we can see the results for each case. We see that only one case out of the 14 gets a majority vote that the report presentation should be other than what the CBR system predicted. The case in question got the predicted presentation *RailText* (can be seen in Table 6.22) from the CBR system but two out of three experts would prefer a pure *Text* presentation instead. Overall 10 out of 42 individual evaluations indicated that another presentation would be preferred but in most of those cases that was the opinion of the minority of the evaluators for a case. As was discussed in Section 6.4.2, 9 out of the 14 cases get the predicted solution *RailText* from the CBR system. Here we need to be aware of the possibility for a bias with regard to the solution chosen by the CBR system, as the evaluators are informed what presentation was chosen.

As was discussed in Section 6.4.2, the ratio of *RailText* predictions was significantly higher than the other, so that presentation was the chosen presentation in 64,3% of the cases. That results in *RailText* not being given as an option for an alternative presentation to choose from in those cases. This partly explains why the *RailText* presentation was never selected in Table 6.25 instead of the chosen presentation for a case.

Now we have a confirmed best solution for all of the experimental case. The solution predicted by the CBR is considered the best one for all but one, which is case 28, where Text presentation is rather preferred by 2 out of three evaluators. We calculate the Pearson correlation coefficient for all presentation versions in relation to the case properties. More correlation is present in this case set than was present in the case set evaluated in phase 1 and discussed in Section 6.2.8. We now have three combinations with correlation of 0.68 or more. *TimeText* and 'Anomalous IDCE' have a correlation of 0.68,*RailText* and 'Count route Request' have and correlation of 0.78, finally, *RailText* and 'Unexpected Route state' have a correlation of 0.87. Considerable negative correlation is also detected between *RailText* and 'Element Request count' (-0.69) and between *Text* and 'Unexpected Route state' (-0.63). Other combinations have a less significant correlation coefficients, varying between -0.6 and 0.6. We calculate the correlation between anomaly types and presentation solutions as well, there we see a correlation of 0.65 between *Text* and anomaly type 4. The rest has a less significant correlation.

Case	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**
Would not prefer other	3	3	2	3	2	2	2	2	2	3	2	2	3	1
Text	X	X	-	-	1	-	1	1	-	-	-	1	X	2
TimeText	-	-	1	-	-	1	X	X	1	-	1	-	-	-
RailText	-	-	X	X	X	X	-	-	X	X	X	X	-	X

Table 6.25: Question 2.3: Measuring the performance of CBR by asking if another presentation version would be rather preferred over the chosen presentation. Each case is evaluated thee times by different experts. The numbers represent a count for each option being chosen. The X denotes the presentation by the CBR system for each case.

Since in only 1 out of 14 cases, or 7% of the cases, the CBR system wrongly predicted the best presentation version for the case, the hypothesis is supported by our experiments.

H6: The report presentation chosen by the CBR system helps the experts understand the anomalous incidents

The evaluation results for all questions contributing to answering this hypothesis can be found in Tables 6.26 and 6.27. They all evaluate the fitness of the report presentation chosen by the CBR system for each case.

In Table 6.26 we can see the results to whether the experts would prefer to use the report presentation chosen by the CBR system over raw log files. Overall 44.2% said yes, 31.0% were not sure and 23.8% would not use the chosen presentation over the raw log files. We run a Cochran's Q test to determine whether there is a significant difference between the answer groups (Cochrans's Q =3,p = 0.223), which does not show a significant difference between the three options.

If we look at each case individually and see what the majority vote was for each of them, 7 cases have a majority of yes, 4 have a majority of maybe, 2 have a majority of no and finally one case has one vote for each option.

Case	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	Sum
Yes	2	2	1	-	1	2	2	2	3	-	-	-	1	3	19
Maybe	-	1	1	2	-	-	1	1	-	1	2	2	2	-	13
No	1	-	1	1	2	1	-	-	-	2	1	1	-	-	10

Table 6.26: Question 2.5: Measuring the usefulness of the reports over raw log files. Each case is evaluated thee times by different experts. The numbers represent a count for each option being chosen.

In Table 6.27 We have the results for two evaluation questions. Question 2.1 measures the *performance* of explaining the anomaly of the chosen report. We have an overall average of 5.86 where the

standard deviation of the evaluations is 1.26. We see that case 19 got the lowest evaluation average out of all cases, which is 4. The standard deviation is highest for that case as well that suggests that the evaluators did not all agree on the performance of that reports. Case 19 is one of the cases that has an inaccuracy in the anomaly description given for the case, which can partly explain why this case got a unusually low evaluation. Two anomaly cases got the highest possible rating from all experts and seven cases got a rating of 6 or more.

Question 2.4 measures the readability of the chosen report. We can see a similar pattern as in the previously discussed question. An average of 5.79 and the exact same standard deviation of 1.26. The lowest evaluation average for a case is 4 and the highest is 7. The lowest evaluation is now for case 17, which was the other anomaly case that had an inaccuracy in the anomaly description.

	Question 2.1			Question 2.4		
Case	**Avg**	**Med**	**Std**	**Avg**	**Med**	**Std**
15	6.0	6.0	1.0	**7.0**	7.0	0.0
16	6.33	6.0	0.58	6.33	7.0	1.15
17	5.33	6.0	2.08	**4.0**	4.0	2.0
18	5.33	5.0	0.58	4.33	4.0	0.58
19	**4.0**	3.0	2.65	4.67	5.0	1.53
20	6.33	6.0	0.58	6.0	6.0	1.0
21	**7.0**	7.0	0.0	6.67	7.0	0.58
22	6.67	7.0	0.58	6.33	7.0	1.15
23	6.33	7.0	1.15	6.33	6.0	0.58
24	4.67	5.0	0.58	5.33	5.0	0.58
25	5.0	5.0	1.0	5.0	5.0	0.0
26	6.0	6.0	0.0	5.0	5.0	0.0
27	6.0	6.0	1.0	**7.0**	7.0	0.0
28	**7.0**	7.0	0.0	**7.0**	7.0	0.0
All	5.86	6.0	1.26	5.79	6.0	1.26

Table 6.27: Results of two evaluation questions asked in evaluation phase 2 on a 7-point rating scale. They both evaluate to the report presentation version chosen by the CBR system. The questions are: Question 2.1: measuring the performance of the reports in explaining the incident and Question 2.4: measuring the readability of the reports. The lowest case average for each question is colored purple and the highest is colored blue.

On the 7-point rating scale, we can define a rating of 4 as a neutral point. Any rating above that is interpreted as considering the report helpful, while ratings below the neutral point indicate unhelpfulness. the average rating for questions 2.1 and 2.4 are well above the neutral point. Moreover, all ratings for every case are at least equal to this neutral point.

This results in all of the three evaluation question showing that the chosen presentations by our CBR system are helpful for the experts to understand anomalous cases in a TCS. These results support H6.

Meeting requirements

Here we will look into the results of question 2.2 which looks into the ability of the output of the system to meet the defined requirements defined for each anomaly type. This is not asked to answer a specific hypothesis, but to measure the competence of the system to meet the requirements defined in Section 4.1.

The results for this question can be found in Table 6.28. We yield an average of 5.83 over all anomalous cases with a standard deviation of 1.06. Cases 19 and 24 get the lowest rating regarding meeting the requirements of their anomaly type. They both have an average over 4 but the standard deviations of their evaluation suggests that they do not fully meet their requirements.

Other Insights

Lastly the open question provided various insightful feedback. The answers mostly contained reasons for low evaluations or additional implementation ideas for the report generation. Those ideas and requirements are good ideas for future work on the project. Eight open comments were given. We will explain the essence of the content of each of them. One answer helped us identify why case 24

	Question 2.2		
Case	**Avg**	**Med**	**Std**
15	6.0	6.0	1.0
16	5.67	6.0	0.58
17	5.67	5.0	1.15
18	5.67	5.0	1.15
19	4.67	6.0	2.31
20	6.33	6.0	0.58
21	**6.67**	7.0	0.58
22	6.0	6.0	0.0
23	**6.67**	7.0	0.58
24	**4.33**	4.0	0.58
25	6.0	6.0	1.0
26	5.67	6.0	0.58
27	5.67	5.0	1.15
28	**6.67**	7.0	0.58
All	5.83	6.0	1.06

Table 6.28: Question 2.2: measuring how well the reports meet the requirements of each incident type. It evaluates to the report presentation version chosen by the CBR system on a 7-point rating scale. The lowest case average is colored purple and the highest is colored blue.

did not perform as well as the average case and furthermore we got a confirmation that the anomaly descriptions for cases 17 and 19 were not correct. That confirmation is very positive as it confirms to us that the reports explain the behaviour of the system well enough for the experts to detect that the anomalous description accompanying the reports did not fit.

The following are all the summarized answers which we have slightly grouped together depending on their content.

- Asking for data that is not included in the log files of interest.

  1. One comment regarded a signal missing from an alerts showing the stop of a ASTRIS software component. This signal is not present in the raw data so it could not have been added.

- Ideas for adjustment of requirements.

  2. In an anomalous case showing a restart of ASTRIS, one expert wanted to see the status of the system and its requests after the restart. That was outside the defined time frame of the anomalies so that was not included, but would be a possible configure if the report generation would be configurable for each case.

- Comments reporting wrong anomaly descriptions.

  3-4. One comment for each of the cases that had slightly incorrect anomaly descriptions(cases 17 and 19) noted that the reports did not explain the anomaly described in the description well. This was known and is informative to see that this was detected by the experts.

- Comments reporting some unclear parts of the reports.

  5. The meaning of a route to be considered unfinished needs more explanation in the reports. A expert thought that meant that something was wrong with the route request, which is not the case.
  6. The behaviour of the system was explained as it was but that did not help the expert understand why a request was denied. That would require more domain specific information and explanations about the expected behaviour of the request, which was not in the scope of the project.

- Programming fault in report for edge case.

7. One comment for case 24 rightfully reported a bug that some important BevNL entries were missing in the report. That resulted from the fact that no IDCE entries were present for a specific elements, en because of that the two BevNL entries for that same element were not included in the report. It is the general rule that IDCE entries are always present for an element if there are BevNL entries. This was not the case in this edge case and should be accounted for in future work on the project of that will be carried out. This explained why case 24 got amongst the lowest evaluations of the experimental cases evaluated in phase 2.

- Lastly, one comment gave a praise.

8. There, the railway layout is said to be helpful for understanding a case by knowing the position of an faulty element

These comments were given in 8 out of 42 possibilities for an open comment. So we assume that in the other cases the experts did not have a specific comment they wanted to depict regarding the case in question.

7 Conclusion

Here we will summarize the findings of our work that have been described in this thesis. We summarize the conclusions we can draw from our evaluation. We will reflect on the limitations of our system design and what would be interesting as the next steps in related research.

In this project we have discussed relevant literature, introduced ASTRIS, the TCS of interest and other domain specific rules and information. We have performed an initial study that provided us with a deeper understanding for the needs of generated reports. We designed and implemented a system that processed multiple types of raw log files into three versions of human readable anomaly reports explaining detected anomalies enhanced with graphical components. The versions of the reports have been evaluated and a CBR system has been designed and implemented to predict the best report version for each anomalous case.

The general feedback we have gotten from our evaluators has been informative and motivating. We have seen that by creating reports with the system in its current condition, we do manage to assist the experts in understanding anomalous incidents. This means that we are successful in processing the raw data into a structured report format. However, there are still numerous features that could be incorporated and would be interesting to see their effects on the performance of the reports.

We will begin by answering our research sub-questions and main research question in Section 7.1. We present the contributions of our work in Section 7.2. Following that, we discuss the limitations of our system and its evaluation in Section 7.3. We then conclude the thesis by presenting interesting future work related to our research in Section 7.4.

7.1. Answering research questions

Research sub-questions:

1. **What report presentation performs best in terms of perceived preference and how does it depend on the properties of the anomalous incident?**

   This research question is answered by three of our hypotheses from evaluation phase 1, H1, H2 and H3. H1 and H2 are not supported since graphical components do not always increase perceived preference and perceived understanding. However, we do see a statistical significance showing that *RailText* overall outperforms the *Text* and *TimeText* presentation versions in terms of perceived preference and perceived understanding.

   We have seen that no single presentation is best for all anomalous cases, therefore, H3 is supported. The cases are all very diverse and have different properties. No clear, decisive trend can be found between the case properties and the report presentation version that provides the highest information quality for each case. To get more conclusive results it would be informative to include more anomalous cases to evaluate, have a larger pool of evaluators and look into more detailed or different case properties.

2. **To what extent do expert preferences of report presentations associate with their expert profiles in relation to data familiarity and work experience?**

   This research question is answered with one hypothesis from evaluation phase 1, H4. H4 is not supported because the preference for report presentations is not associated with the work experience of the evaluators. However, we identified an association between the expert's log familiarity and their perceived preferences of the report presentations. We see that experts that have low log familiarity do rather prefer reports containing graphical components over purely textual reports. On the other hand, results for experts with high log familiarity do not show a significant preference for any one presentation version.

3. **How is Case-Based Reasoning, as an adaptive template selection method for anomalous incidents, where incidents are represented with variables, perceived by experts?**

   This research question is answered with two hypotheses from evaluation phase 2, H5 and H6. They are both supported by our experiments. The output of our CBR system is well received, 13 out of 14 cases are considered rightfully predicted by the experts. The chosen presentation helps experts understand anomalous incidents. We get good average results of 5.86 out of 7 for how well the chosen presentation explains the incident and the average of the readability is 5.79. The averages over all chosen presentation for these questions are higher than any single presentation yielded for the same question in evaluation phase 1, which suggest that the CBR system manages to improve the overall readability and performance of the reports.

Main RQ:

When generating anomalous incident reports in a train control system, how should presentation formats be adaptively selected for incidents to provide the highest perceived information quality for domain experts?

We conclude that the properties of anomalous cases and experts' familiarity with the data should be taken into account when generating anomalous reports in a TCS. Presentation formats should be adaptively selected due to the diverse nature of anomalous incidents. We have shown that a CBR system can perform this task of adaptively selection presentation formats. The CBR system is perceived well when used to select presentation versions for new unseen anomalous cases. This should be done by using the properties of anomalous cases to detect similarity between cases. Cases with a similar composition of properties should be explained using the same report presentation format.

The inclusion of graphical components showing an annotated railway overview generally increases perceived information quality. The inclusion of the timeline component can sometimes yield high information quality but is dependent on the properties of the anomalous cases. Data familiarity of experts influences the perception reports and should therefore be taken into account in the presentation selection to further increase the effectiveness of the reports.

7.2. Contributions

We have presented a data-to-text, report generation system that can generate three different versions of reports. Each version accounts for a specific trait of the data in question by including two types of graphical components. We automatically generate these novel, annotated, graphical components, that dynamically report information that is relevant for a specific anomalous case. This system performs complex content selection procedures on raw log entries, which depend on complex, domain-specific rules.

We have shown that graphical components in explanatory reports can improve understanding and readability. We have also designed a CBR system inspired by methodologies from literature for a new domain. We have demonstrated that it can be applied to the task of predicting report presentations for anomalous cases depending on their properties. This application of CBR is novel in the sense that graphical components are adaptively included according to case properties.

The significance of the system designed and implemented in this thesis is that it can assist the domain experts of the TCS to understand anomalous incidents that have occurred. Furthermore, it should decrease the need for extensive manual log analysis and thus save time and help make the anomaly explanation process more effective. We have shown that it is feasible to apply our report

generation methodology to the complex domain of TCS, where we have incorporated multiple log file types with ill-defined relationships. We believe that our work could be extended to other similar, complex domains. To do so, the domain-specific aspects of the system need to be reformulated to fit the new domain. However, we provide evidence into what types of report presentations are most useful to domain experts.

7.3. Limitations

The domain of interest requires the reports to be domain-specific and its evaluators to be very familiar with the domain. This single domain application can be considered as a limitation and it would be very interesting to apply a similar methodology to a related domain, which is not as expert specific.

The research has provided some valuable insights towards answering our research questions. However, the small and uniform pool of evaluators, suitable for evaluating the system results in less evaluation power. Since the number of evaluators per case is not high, we can not conclude that the results are always reflective of the opinion of the whole expert pool. This low number of evaluators also resulted in the number of possible evaluations to perform being low. The current evaluation setting does not allow us to draw a strong, unconditional conclusion. It would be preferable to have more experts that reflect a broader spectrum of properties to evaluate more cases. Despite having a limited evaluation setup due to the reasons discussed here, we still perform human-based evaluation using mostly extrinsic evaluation measures. This kind of setup is generally considered better than the more frequently used automatic and intrinsic evaluation methods[59].

Some aspects of the generation and application of graphical components have caused some limitations in our work. The graphical components can get overly cluttered when many repetitions of requests take place. Thus, the timeline components become difficult to read. This could be improved with a more detailed and dynamic design for the timelines or splitting the timelines into multiple timelines. In cases where there is little data, the timelines on the other hand are relatively empty and unhelpful, so in cases like that, it would be interesting to combine multiple timelines.

Furthermore, the railway overview generation still requires some manual work, to extract and mark positions of elements. This could be improved by making use of automatic text detection to identify the location of elements in the raw railway overview. With these locations, the railway annotation could be completely automatic.

7.4. Future Work

The work we have done in this thesis provides the foundation for all parts of the report generation pipeline. Some parts can be improved with further research and time. The matter of explaining anomalies from these log files could be researched extensively for a few years and you would still have interesting aspects to look into. The following are some interesting possible extensions of the NLG system.

- We could add more types of data to allow for an even more detailed explanation of the behavior of the system leading to anomalous behavior.
- We could include active monitoring of live generation of log files to research the effects of the system in a real-life scenario.
- A more elegant NLG methodology, moving from templates to a more dynamic setup, would be interesting to incorporate where more domain-specific linguistic rules could be defined and implemented. That would however require the creation of a set of a domain-specific language base.
- The system implemented in this thesis is solely intended for domain experts. It would be interesting to extend this work to make it understandable for the operators of the TCS, who may not have as much detailed knowledge of the system and may have different requirements.

The next step in terms of the prediction of templates for the generation of cases would be to compare the effectiveness of CBR to other prediction systems that can account for diverse properties. For example more traditional classification algorithms. We focused on CBR due to its ability to incorporate the revision phase where expert evaluation can influence the construction of the case base.

Furthermore, the incorporation of other case properties would be interesting to model the anomalous cases, such as the density of requests instead of their count. Different weights could be applied to different case properties within the CBR so that some properties have more impact on the final decision than others. The importance of each feature can be evaluated by domain experts to gain good insights.

Lastly, it would be interesting to account for the different case properties in an interactive system. Where the system can propose a good solution for an anomalous case, but the expert can incorporate any specific needs or preferences for the specific report. The experts can configure what combination of graphical components they want in a report and specify exactly what data they want to include.

Bibliography

[1] Building cbr systems with jcolibri. *Science of Computer Programming*, 69(1):68 – 75, 2007. ISSN 0167-6423. doi: https://doi.org/10.1016/j.scico.2007.02.004. URL http://www.sciencedirect.com/science/article/pii/S0167642307001645. Special issue on Experimental Software and Toolkits.

[2] Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, March 1994. ISSN 0921-7126. URL http://dl.acm.org/citation.cfm?id=196108.196115.

[3] Ibrahim Adeyanju. Generating weather forecast texts with case based reasoning. *International Journal of Computer Applications*, 45:35–40, 05 2012. doi: 10.5120/6819-9176.

[4] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, May 2015. ISSN 1573-756X. doi: 10.1007/s10618-014-0365-y. URL https://doi.org/10.1007/s10618-014-0365-y.

[5] Jose Alonso, Patricia Conde-Clemente, and Gracian Trivino. Linguistic description of complex phenomena with the rLDCP r package. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 243–244, Santiago de Compostela, Spain, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-3538. URL https://www.aclweb.org/anthology/W17-3538.

[6] Jacopo Amidei, Paul Piwek, and Alistair Willis. Agreement is overrated: A plea for correlation to assess human evaluation reliability. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 344–354, Tokyo, Japan, October - November 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W19-8642.

[7] Jacopo Amidei, Paul Piwek, and Alistair Willis. The use of rating and Likert scales in natural language generation human evaluation tasks: A review and some recommendations. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 397–402, Tokyo, Japan, October - November 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W19-8648.

[8] Bert Bossink. System/subsystem design description astris. Technical Report 8.5, ICT Services, 1 2019. ASTRIS-SSDD.

[9] Daniela Briola, Riccardo Caccia, Michele Bozzano, and Angela Locoro. Ontologica: Exploiting ontologies and natural language for representing and querying railway management logics. volume 243, pages 376–385, 01 2012. doi: 10.3233/978-1-61499-105-2-376.

[10] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep. pages 891–900, 10 2015. doi: 10.1145/2806416.2806512.

[11] Sungbin Cho, Hyojung Hong, and Byoung-Chun Ha. A hybrid approach based on the combination of variable selection using decision trees and case-based reasoning using the mahalanobis distance: For bankruptcy prediction. *Expert Systems with Applications*, 37:3482–3488, 04 2010. doi: 10.1016/j.eswa.2009.10.040.

[12] William Jay Conover. *Practical nonparametric statistics*. Wiley series in probability and statistics. Wiley, New York, NY [u.a.], 3. ed edition, 1999. ISBN 0471160687. URL http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+24551600X&sourceid=fbw_bibsonomy.

[13] Kees van Deemter, Mariët Theune, and Emiel Krahmer. Real versus template-based natural language generation: A false opposition? *Computational Linguistics*, 31(1):15–24, 2005. doi: 10.1162/0891201053630291. URL https://doi.org/10.1162/0891201053630291.

[14] Alessandro Fantechi, Francesco Flammini, and Stefania Gnesi. Formal methods for railway control systems. *International Journal on Software Tools for Technology Transfer*, 16(6):643–646, Nov 2014. ISSN 1433-2787. doi: 10.1007/s10009-014-0342-1. URL https://doi.org/10.1007/s10009-014-0342-1.

[15] Alessio Ferrari, Maurice ter Beek, Franco Mazzanti, Davide Basile, Alessandro Fantechi, Stefania Gnesi, Andrea Piattino, and Daniele Trentini. *Survey on Formal Methods and Tools in Railways: The ASTRail Approach*, pages 226–241. 01 2019. ISBN 978-3-030-18743-9. doi: 10.1007/978-3-030-18744-6_15.

[16] Albert Gatt and Emiel Krahmer. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *CoRR*, abs/1703.09902, 2017. URL http://arxiv.org/abs/1703.09902.

[17] D. Gkatzia, O. Lemon, and V. Rieser. Data-to-text generation improves decision-making under uncertainty. *IEEE Computational Intelligence Magazine*, 12(3):10–17, 2017.

[18] Rob M.P. Goverde and Lingyun Meng. Advanced monitoring and management information of railway operations. *Journal of Rail Transport Planning and Management*, 1(2):69 – 79, 2011. ISSN 2210-9706. doi: https://doi.org/10.1016/j.jrtpm.2012.05.001. URL http://www.sciencedirect.com/science/article/pii/S2210970612000029.

[19] Dina Hadžiosmanović, Damiano Bolzoni, and Pieter H. Hartel. A log mining approach for process monitoring in scada. *International Journal of Information Security*, 11(4):231–251, Aug 2012. ISSN 1615-5270. doi: 10.1007/s10207-012-0163-8. URL https://doi.org/10.1007/s10207-012-0163-8.

[20] Xiao Han, Tao Tang, Jidong Lv, and Haifeng Wang. Failure analysis of chinese train control system level 3 based on model checking. In Thierry Lecomte, Ralf Pinger, and Alexander Romanovsky, editors, *Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification*, pages 95–105, Cham, 2016. Springer International Publishing. ISBN 978-3-319-33951-1.

[21] Raquel Hervás and Pablo Gervás. Case-based reasoning for knowledge-intensive template selection during text generation. volume 4106, pages 151–165, 09 2006. doi: 10.1007/11805816_13.

[22] Saar Hommes, Chris van der Lee, Felix Clouth, Jeroen Vermunt, Xander Verbeek, and Emiel Krahmer. A personalized data-to-text support tool for cancer patients. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 443–452, Tokyo, Japan, October - November 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W19-8656.

[23] Breier Jakub and Jana Branišová. Anomaly detection from log files using data mining techniques. *Lecture Notes in Electrical Engineering*, 339:449–457, 01 2015. doi: 10.1007/978-3-662-46578-3_53.

[24] Pamela Jordan, Nancy Green, Chistopher Thomas, and Susan Holm. TBI-doc: Generating patient & clinician reports from brain imaging data. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, pages 143–146, Philadelphia, Pennsylvania, U.S.A., June 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-4423.

[25] Pavle Kecman and Rob Goverde. Process mining of train describer event data and automatic conflict identification. 01 2013. doi: 10.2495/CR120201.

[26] T. Khoshgoftaar, Naeem Seliya, and Nandini Sundaresh. An empirical study of predicting software faults with case-based reasoning. *Software Quality Journal*, 14:85–111, 2006.

[27] Shahid Latif and Fabian Beck. Vis author profiles: Interactive descriptions of publication records combining text and visualization. *IEEE Transactions on Visualization and Computer Graphics*, PP: 1–1, 08 2018. doi: 10.1109/TVCG.2018.2865022.

[28] Shahid Latif and Fabian Beck. Interactive map reports summarizing bivariate geographic data. *Visual Informatics*, 3(1):27 – 37, 2019. ISSN 2468-502X. doi: https://doi.org/10.1016/j.visinf.2019.03.004. URL http://www.sciencedirect.com/science/article/pii/S2468502X19300191. Proceedings of PacificVAST 2019.

[29] Anna Law, Yvonne Freer, Jim Hunter, Robert Logie, Neil Mcintosh, and John Quinn. A comparison of graphical and textual presentations of time series data to support medical decision making in the neonatal intensive care unit. *Journal of clinical monitoring and computing*, 19:183–94, 07 2005. doi: 10.1007/s10877-005-0879-3.

[30] Yves Ledru, Akram Idani, Rahma Ben Ayed, Abderrahim Ait Wakrime, and Philippe Bon. A separation of concerns approach for the verified modelling of railway signalling rules. In Simon Collart-Dutilleul, Thierry Lecomte, and Alexander Romanovsky, editors, *Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification*, pages 173–190, Cham, 2019. Springer International Publishing. ISBN 978-3-030-18744-6.

[31] Saad Mahamood, William Bradshaw, and Ehud Reiter. Generating annotated graphs using the NLG pipeline architecture. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, pages 123–127, Philadelphia, Pennsylvania, U.S.A., June 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-4419. URL https://www.aclweb.org/anthology/W14-4419.

[32] Joy Mahapatra, Sudip Kumar Naskar, and Sivaji Bandyopadhyay. Statistical natural language generation from tabular non-textual data. In *Proceedings of the 9th International Natural Language Generation conference*, pages 143–152, Edinburgh, UK, September 5-8 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-6624. URL https://www.aclweb.org/anthology/W16-6624.

[33] Alessandro Mazzei. Translating Italian to LIS in the rail stations. In *Proceedings of the 15th European Workshop on Natural Language Generation (ENLG)*, pages 76–80, Brighton, UK, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-4712. URL https://www.aclweb.org/anthology/W15-4712.

[34] Martin Molina, Amanda Stent, and Enrique Parodi. Generating automated news to explain the meaning of sensor data. In *Proceedings of the 10th International Conference on Advances in Intelligent Data Analysis X*, IDA'11, pages 282–293, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-24799-6. URL http://dl.acm.org/citation.cfm?id=2075337.2075366.

[35] Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1238. URL https://www.aclweb.org/anthology/D17-1238.

[36] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2001.

[37] Rivindu Perera and Parma Nand. Recent advances in natural language generation: A survey and classification of the empirical literature. *Computing and Informatics*, 36:1–32, 2017.

[38] Vassilis Plachouras, Charese Smiley, Hiroko Bretz, Ola Taylor, Jochen L. Leidner, Dezhao Song, and Frank Schilder. Interacting with financial data using natural language. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 1121–1124, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4069-4. doi: 10.1145/2911451.2911457. URL http://doi.acm.org/10.1145/2911451.2911457.

[39] François Portet, Ehud Reiter, Albert Gatt, Jim Hunter, Somayajulu Sripada, Yvonne Freer, and Cindy Sykes. Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence*, 173(7):789 – 816, 2009. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2008.12.002. URL http://www.sciencedirect.com/science/article/pii/S0004370208002117.

[40] Ehud Reiter. An architecture for data-to-text systems. In *Proceedings of the Eleventh European Workshop on Natural Language Generation (ENLG 07)*, pages 97–104, Saarbrücken, Germany, June 2007. DFKI GmbH. URL https://www.aclweb.org/anthology/W07-2315.

[41] Ehud Reiter and Rober Dale. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87, 1997. doi: 10.1017/S1351324997001502.

[42] Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge University Press, New York, NY, USA, 2000. ISBN 0-521-62036-8.

[43] R.H. Riffenburgh. Chapter 11 - tests on ranked data. In R.H. Riffenburgh, editor, *Statistics in Medicine (Third Edition)*, pages 221 – 248. Academic Press, San Diego, third edition edition, 2012. ISBN 978-0-12-384864-2. doi: https://doi.org/10.1016/B978-0-12-384864-2.00011-1. URL http://www.sciencedirect.com/science/article/pii/B9780123848642000111.

[44] Frank Ruessink. System/subsystem specification astris. Technical Report 7.1, ICT Services, 1 2019. ASTRIS-SSS.

[45] Joe Saliby. Survey on natural language generation. *International Journal of Trend in Scientific Research and Development*, Volume-3:618–622, 04 2019. doi: 10.31142/ijtsrd22903.

[46] Roger C. Schank. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, USA, 1983. ISBN 0521248582.

[47] Martin Schiersch, Veselina Mironova, Maximilian Schmitt, Philippe Thomas, Aleksandra Gabryszak, and Leonhard Hennig. A German corpus for fine-grained named entity recognition and relation extraction of traffic and industry events. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan, May 2018. European Languages Resources Association (ELRA). URL https://www.aclweb.org/anthology/L18-1703.

[48] Gleb Sizov and Pinar Öztürk. Automatic extraction of reasoning chains from textual reports. In *Proceedings of TextGraphs-8 Graph-based Methods for Natural Language Processing*, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W13-5009.

[49] Arjun Srinivasan, Steven M. Drucker, Alex Endert, and John T. Stasko. Augmenting visualizations with interactive data facts to facilitate interpretation and communication. *IEEE Transactions on Visualization and Computer Graphics*, 25:672–681, 2019.

[50] Somayajulu Sripada. Summarizing dive computer data: A case study in integrating textual and graphical presentations of numerical data. 2007.

[51] Y. Tagawa and K. Shimada. Generating abstractive summaries of sports games from japanese tweets. In *2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, pages 82–87, July 2016. doi: 10.1109/IIAI-AAI.2016.166.

[52] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, Fourth Edition*. Academic Press, Inc., USA, 4th edition, 2008. ISBN 1597492728.

[53] Jesse Thomason, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Raymond Mooney. Integrating language and vision to generate natural language descriptions of videos in the wild. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1218–1227, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics.

[54] Nava Tintarev, Yolanda Melero, Somayajulu Sripada, Elizabeth Tait, Rene Van Der Wal, and Chris Mellish. Minkapp: generating spatio-temporal summaries for nature conservation volunteers. pages 71–21. ACL Association for Computational Linguistics, 2012. URL http://hdl.handle.net/10059/1878. COMPLETED – Requested permission from acl@org.aclweb, but email address undeliverable. 23/9/2016 LM – Info from contact 21/9/2016 LM ADDITIONAL INFORMATION: Tait, Elizabeth.

[55] Khoa Tran and Fred Popowich. Automatic tweet generation from traffic incident data. In *Proceedings of the 2nd International Workshop on Natural Language Generation and the Semantic Web (WebNLG 2016)*, pages 59–66, Edinburgh, Scotland, September 2016. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W16-3512.

[56] Chris Turner, Ashutosh Tiwari, Andrew Starr, and Kevin Blacktop. A review of key planning and scheduling in the rail industry in europe and uk. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, 230:984–998, 02 2016. doi: 10.1177/0954409714565654.

[57] Ross Turner, Somayajulu Sripada, Ehud Reiter, and Ian Davy. Using spatial reference frames to generate grounded textual summaries of georeferenced data. In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 16–24, Salt Fork, Ohio, USA, June 2008. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W08-1104.

[58] Bojana Urumovska. From log files to train traffic reports : Using Natural Language Generation to explain anomalies from Train Control System log files. Master's thesis, Technical university of Delft, the Netherlands, 2019.

[59] C van der Lee, A Gatt, E van Miltenburg, S Wubben, and E Krahmer. Best practices for the human evaluation of automatically generated text. In *Proceedings of the 12th International Conference on Natural Language Generation (INLG'19)*, Tokyo, Japan, 2019. Association for Computational Linguistics. URL https://www.inlg2019.com/assets/papers/98_Paper.pdf.

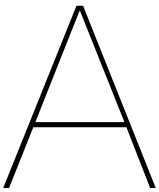
[60] Chris van der Lee, Emiel Krahmer, and Sander Wubben. PASS: A Dutch data-to-text system for soccer, targeted towards specific audiences. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 95–104, Santiago de Compostela, Spain, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-3513. URL https://www.aclweb.org/anthology/W17-3513.

[61] Chris van der Lee, Emiel Krahmer, and Sander Wubben. Automated learning of templates for data-to-text generation: comparing rule-based, statistical and neural methods. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 35–45, Tilburg University, The Netherlands, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6504. URL https://www.aclweb.org/anthology/W18-6504.

[62] Piek Vossen, Filip Ilievski, Marten Postma, and Roxane Segers. Don't annotate, but validate: a data-to-text method for capturing event data. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan, May 2018. European Languages Resources Association (ELRA). URL https://www.aclweb.org/anthology/L18-1480.

[63] Judith C. Wagner, Jeremy E. Rogers, Robert H. Baud, and Jean-Raoul Scherrer. Natural language generation of surgical procedures. *International Journal of Medical Informatics*, 53(2):175 – 192, 1999. ISSN 1386-5056. doi: https://doi.org/10.1016/S1386-5056(98)00158-0. URL http://www.sciencedirect.com/science/article/pii/S1386505698001580.

[64] Huanyu Yu, Shuo Cheng, Bingbing Ni, Minsi Wang, Jian Zhang, and Xiaokang Yang. Fine-grained video captioning for sports narrative. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.



Detailed evaluation results

	Presentation	First place	Second place	Third place
Case0	Text	2	-	1
	TimeText	1	-	2
	RailText	-	3	-
Case1	Text	-	-	3
	TimeText	2	1	-
	RailText	1	2	-
Case2	Text	-	1	2
	TimeText	2	-	1
	RailText	1	2	-
Case3	Text	1	2	-
	TimeText	-	1	2
	RailText	2	-	1
Case4	Text	-	1	2
	TimeText	3	-	-
	RailText	-	2	1
Case5	Text	1	1	1
	TimeText	1	-	2
	RailText	1	2	-
Case6	Text	1	2	-
	TimeText	-	1	2
	RailText	2	-	1
Case7	Text	-	-	3
	TimeText	1	2	-
	RailText	2	1	-
Case8	Text	-	2	1
	TimeText	1	-	2
	RailText	2	1	-
Case9	Text	1	1	1
	TimeText	1	-	2
	RailText	1	2	-
Case10	Text	2	-	1
	TimeText	1	-	2
	RailText	-	3	-
Case11	Text	1	-	2
	TimeText	1	1	1
	RailText	1	2	-
Case12	Text	1	1	1
	TimeText	1	-	2
	RailText	1	2	-
Case13	Text	-	2	1
	TimeText	1	-	2
	RailText	2	1	-
Case14	Text	2	1	-
	TimeText	1	2	-
	RailText	-	-	-
All	Text	12 (26.7%)	14 (31.1%)	19 (45.2%)
	TimeText	17 (37.8%)	8 (17.8%)	20 (47.6%)
	RailText	16 (35.5%)	23 (51.1%)	3 (7.2%)

Table A.1: Detailed answers to evaluation question 1.2: “Please order the reports in decreasing order of how well the report helps you understand the anomaly”. Measuring the performance of the reports. Asked in evaluation phase 1.

	Presentation	First place	Second place	Third place
Case0	Text	2	-	1
	TimeText	1	-	2
	RailText	-	3	-
Case1	Text	-	-	3
	TimeText	2	1	-
	RailText	1	2	-
Case2	Text	-	1	2
	TimeText	1	1	1
	RailText	2	1	-
Case3	Text	1	2	-
	TimeText	-	1	2
	RailText	2	-	1
Case4	Text	-	1	2
	TimeText	3	-	-
	RailText	-	2	1
Case5	Text	1	1	1
	TimeText	1	-	2
	RailText	1	2	-
Case6	Text	1	2	-
	TimeText	-	1	2
	RailText	2	-	1
Case7	Text	-	-	3
	TimeText	1	2	-
	RailText	2	1	-
Case8	Text	-	2	1
	TimeText	1	-	2
	RailText	2	1	-
Case9	Text	1	1	1
	TimeText	1	-	2
	RailText	1	2	-
Case10	Text	2	-	1
	TimeText	1	-	2
	RailText	-	3	-
Case11	Text	2	-	1
	TimeText	-	1	2
	RailText	1	2	-
Case12	Text	1	1	1
	TimeText	1	-	2
	RailText	1	2	-
Case13	Text	1	1	1
	TimeText	1	-	2
	RailText	1	2	-
Case14	Text	2	1	-
	TimeText	1	2	-
	RailText	-	-	-
All	Text	14 (31.1%)	13 (28.8%)	18 (42.9%)
	TimeText	15 (33.3%)	9 (20%)	21 (50%)
	RailText	16 (35.6%)	22 (48.8%)	3 (7.1%)

Table A.2: Detailed answers to evaluation question 1.3: “Please order the reports in decreasing order of which report version you would prefer to use to for this anomaly”. Measuring Preference of the report presentations. Asked in evaluation phase 1.

	Presentation	First place	Second place	Third place
Case0	Text	2	-	1
	TimeText	1	-	2
	RailText	-	3	-
Case1	Text	-	1	2
	TimeText	2	1	-
	RailText	1	1	1
Case2	Text	-	1	2
	TimeText	2	-	1
	RailText	1	2	-
Case3	Text	1	2	-
	TimeText	-	-	3
	RailText	2	1	-
Case4	Text	-	1	2
	TimeText	2	1	-
	RailText	1	1	1
Case5	Text	1	1	1
	TimeText	1	-	2
	RailText	1	2	-
Case6	Text	1	2	-
	TimeText	-	1	2
	RailText	2	-	1
Case7	Text	-	-	3
	TimeText	1	2	-
	RailText	2	1	-
Case8	Text	-	1	2
	TimeText	1	1	1
	RailText	2	1	-
Case9	Text	1	1	1
	TimeText	-	1	2
	RailText	2	1	-
Case10	Text	1	1	1
	TimeText	-	1	2
	RailText	2	1	-
Case11	Text	1	1	1
	TimeText	-	2	1
	RailText	2	-	1
Case12	Text	1	1	1
	TimeText	1	-	2
	RailText	1	2	-
Case13	Text	-	2	1
	TimeText	1	-	2
	RailText	2	1	-
Case14	Text	2	1	-
	TimeText	1	2	-
	RailText	-	-	-
All	Text	11(24.4%)	16 (35.5%)	18 (42.9%)
	TimeText	13 (28.9%)	12 (26.7%)	20 (47.6%)
	RailText	21 (46.7%)	17 (37.8%)	4 (9.5%)

Table A.3: Detailed answers to evaluation question 1.4: “Please order the reports in decreasing order according of how well the report represents the system”. Measuring the performance of the reports to represent the system. Asked in evaluation phase 1.

B

Example Reports

Here we can find example reports that show all of our presentation versions. The reports do not show information for any specific experimental case handled in the thesis. It is meant show an example of the layout.

We will first include an example of a *Text* report, followed by a *TimeText* example and lastly a *RailText* example.

Example Text Report

The timeframe of interest for this report is from: 2020-07-02T21:19:39Z to: 2020-07-02T21:20:45Z. In this report we will look into active processes during this time frame and report events that are unexpected or could cause trouble.

In this report a red color will be used to highlight anomalous behaviour and yellow color will be to highlight changes between entries.

The following alerts (meldingen) were given within the timeframe.

Timestamp	Melding
21:19:58Z	Opdracht afgekeurd vanuit beveiliging.()

4 BevNL messages were sent between Astris and BevNL during the time period, out of which 2 requests and 2 responses, 1 requests were accepted and 1 declined.

There where 4 ARC entries within the timeframe of interest of which 2 route requests and 2 route related responses, 0 element requests and 0 element related responses.
The 2 requests that were accepted are:
- 1 instellenRijweg_1 for route 106-132RL,
- 1 instellenRijweg_1 for route 154-108LLRR.

Route request processes

We will now look into all routes that have active processes during the timeframe. The requests are grouped by routes. For each route we first go through an overview of the setting an behaviour of the route. Next we look into behaviour of IDCR entries and end with a detailed overview of element attributes.

The following routes had all requests normal and finished within the timeframe:
Route 154-ZL_108-ZL_LLRR
There is nothing alarming regarding these processes so we will not look more into them.

The following routes had some unexpected behaviour:
Route 106-ZL_132-ZL_RL
We will look more closely into them in the next section.

Route : 106-ZL_132-ZL_RL

The behaviour of this route is considered Incorrect for the reasons:
- command with ID GH1_15937247979422 got the value 'NietGeaccepteerd' from Astris in an IDCR entry,
- Rijweg BevNL command InstellenRoute was rejected with reason 'Opdracht afgekeurd vanuit beveiliging',
- request instellenRijweg_1 with id GH1_15937247979422 ended up in an unexpected Rijweg toestand and
- last status of request instellenRijweg_1 with id GH1_15937247979422 is Instelopdracht_afgekeurd.

There is one element in this route that is also part of another route. It is the element SECTIE A108T.

Configuration setup for this route is - NormaalToegestaan: Ja, AutomaatToegestaan: Nee, ROZToegestaan: Ja, STSToegestaan: Ja, Geelectrificeerd: Ja, AstrisMoetVoorbereiden: Ja, ElementGestuurdeBeveiliging: Nee. Furthermore the VoorSectieIdentificatie of the route is: sectie A86T.

BevNL entries for this Rijweg

1 requests got rejected by the BEVNL layer. The requests that got rejected are InstellenRoute with reason 'Opdracht afgekeurd vanuit beveiliging'.

Timestamp	Command	CorrelatieId	BeginSein	EindSein	Acceptatie	AfkeurRedenNr	AfkeurRedenTekst
21:19:58.1284	InstellenRoute	5	SEIN-106-ZLPPLG-1	SEIN-132-ZLPPLG-1			
21:19:58.139Z	Antwoord	5	----	----	N	1	Opdracht afgekeurd vanuit beveiliging.

Requests for this route

Requests instellenRijweg_1

Some behaviour was considered unexpected for this request, that is because of:
- command with ID GH1_15937247979422 got the value 'NietGeaccepteerd' from Astris in an IDCR entry
- request instellenRijweg_1 with id GH1_15937247979422 ended up in an unexpected Rijweg toestand
- last status of request instellenRijweg_1 with id GH1_15937247979422 is Instelopdracht_afgekeurd

The ARC for this entry was at time 2020-07-02T21:19:57.9752Z, with action instellenRijweg_1 and opdrachtIdentifiatie GH1_15937247979422 with Seinbediening: ROZ .

IDCR entries for the status of the route

Timestamp	OpdrachtID	Attributes	ToestandRijweg
21:19:58.0694	GH1_15937247979422	RR IR	Rust
21:19:58.0857	GH1_15937247979422	IR	Gereserveerd
21:19:58.0998	GH1_15937247979422	IR	Voorbereid
21:19:58.1400	GH1_15937247979422		Instelopdracht_verstuurd
21:19:58.1609	GH1_15937247979422		Instelopdracht_afgekeurd

IDCR entries for the status of the request

Timestamp	OpdrachtID	OpdrachtToestand	Acceptatie
21:19:57.9802	GH1_15937247979422	Controleren	
21:19:58.0686	GH1_15937247979422	Uitvoeren	
21:19:58.1616	GH1_15937247979422	Gereed	NietGeaccepteerd

Elements in, by or connected to this route

We will now go in detail into the state transition of the elements that are part of the current route of interest.

Elements in this route

Entries for WISSEL-105B-ZL-PPLG-1

The element is only part of this active route process during the timeframe

Overview of IDCE entries for the element

Timestamp	OpdrachtID	Rijweg	ReqStand	Gebruikt	StandLRT	SturingLR	LigtL	LigtR	Locked	Betrouwbaar	NietInControle	OpenZonderSturing
21:19:53.3795				false		L	true	false	false	true	false	false

Overview of BevNL entries for the element

Timestamp	ABR_Links_waarde	ABR_Rechts_waarde	Betrouwbaar	LigtLinks	LigtRechts	Sturing	Locked	NietInControle	OpenZonderSturing	TRDL_VHB	Stand	Type of entry	Reply
21:19:53.364Z	N	N		true	false	L	false	false	false	false		Obj Rep	

Entries for SECTIE-107AT-ZL-PPLG-1

The element is only part of this active route process during the timeframe

Overview of IDCE entries for the element

Timestamp	OpdrachtID	Rijweg	Gebruikt	Bezet	LigtInRijweg	RijwegRv.	GebiedRv.	Betrouwbaar
21:19:58.0535	GH1_15937247979422	106-132	false	false	false	true	false	true

There are no BEVNL entries for this element

Elements coupled to this element in this route

Entries for WISSEL-107B-ZL-PPLG-1

The element is only part of this active route process during the timeframe

Overview of IDCE entries for the element

Timestamp	OpdrachtID	Rijweg	ReqStand	Gebruikt	StandLRT	SturingLR	LigtL	LigtR	Locked	Betrouwbaar	NietInControle	OpenZonderSturing
21:19:53.3749				false		R	false	true	false	true	false	false

Overview of BevNL entries for the element

Timestamp	ABR_Links_waarde	ABR_Rechts_waarde	Betrouwbaar	LigtLinks	LigtRechts	Sturing	Locked	NietInControle	OpenZonderSturing	TRDL_VHB	Stand	Type of entry	Reply
21:19:53.363Z	N	N		false	true	R	false	false	false	false		Obj Rep	

Elements in this route that are part of NPV situations

Some of these elements have already been handled for this route. They are:

WISSEL-107A-ZL-PPLG-1

Example TimeText Report

The timeframe of interest for this report is from: 2020-07-02T21:19:39Z to: 2020-07-02T21:20:45Z. In this report we will look into active processes during this time frame and report events that are unexpected or could cause trouble.

In this report a red color will be used to highlight anomalous behaviour and yellow color will be to highlight changes between entries.

The following alerts (meldingen) were given within the timeframe.

Timestamp	Melding
21:19:58Z	Opdracht afgekeurd vanuit beveiliging.()

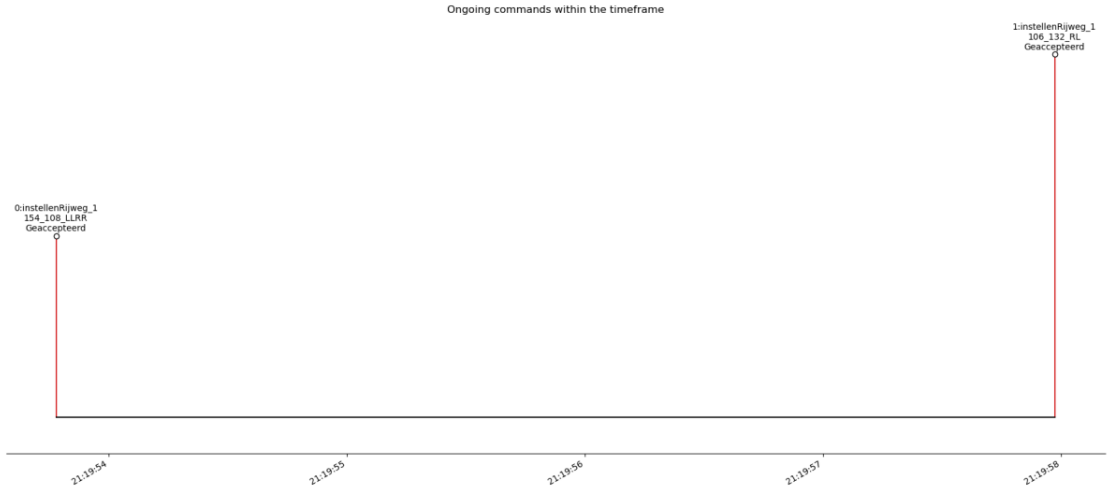
4 BevNL messages were sent between Astris and BevNL during the time period, out of which 2 requests and 2 responses, 1 requests were accepted and 1 declined.

There where 4 ARC entries within the timeframe of interest of which 2 route requests and 2 route related responses, 0 element requests and 0 element related responses.
The 2 requests that were accepted are:

- 1 instellenRijweg_1 for route 106-132RL,
- 1 instellenRijweg_1 for route 154-108LLRR.

On the timelines below we can see how they distribute over time and what response each request got.

These are the ongoing route commands accompanied with their responses according to the ARC entries:



Route request processes

We will now look into all routes that have active processes during the timeframe. The requests are grouped by routes. For each route we first go through an overview of the setting an behaviour of the route. Next we look into behaviour of IDCR entries and end with a detailed overview of element attributes.

The following routes had all requests normal and finished within the timeframe:
Route 154-ZL_108-ZL_LLRR
There is nothing alarming regarding these processes so we will not look more into them.

The following routes had some unexpected behaviour:
Route 106-ZL_132-ZL_RL
We will look more closely into them in the next section.

Route : 106-ZL_132-ZL_RL

The behaviour of this route is considered Incorrect for the reasons:

- command with ID GH1_15937247979422 got the value 'NietGeaccepteerd' from Astris in an IDCR entry,

- Rijweg BevNL command InstellenRoute was rejected with reason 'Opdracht afgekeurd vanuit beveiliging',
- request instellenRijweg_1 with id GH1_15937247979422 ended up in an unexpected Rijweg toestand and
- last status of request instellenRijweg_1 with id GH1_15937247979422 is Instelopdracht_afgekeurd.

There is one element in this route that is also part of another route. It is the element SECTIE A108T.

Configuration setup for this route is - NormaalToegestaan: Ja, AutomaatToegestaan: Nee, ROZToegestaan: Ja, STSToegestaan: Ja, Geelectrificeerd: Ja, AstrisMoetVoorbereiden: Ja, ElementGestuurdeBeveiliging: Nee. Furthermore the VoorSectieIdentificatie of the route is: sectie A86T.

BevNL entries for this Rijweg

1 requests got rejected by the BEVNL layer. The requests that got rejected are InstellenRoute with reason 'Opdracht afgekeurd vanuit beveiliging'.

Timestamp	Command	CorrelatieId	BeginSein	EindSein	Acceptatie	AfkeurRedenNr	AfkeurRedenTekst
21:19:58.1284	InstellenRoute	5	SEIN-106-ZL-PPLG-1	SEIN-132-ZL-PPLG-1			
21:19:58.139Z	Antwoord	5	----	----	N	1	Opdracht afgekeurd vanuit beveiliging.

Requests for this route

Requests instellenRijweg_1

Some behaviour was considered unexpected for this request, that is because of:
- command with ID GH1_15937247979422 got the value 'NietGeaccepteerd' from Astris in an IDCR entry
- request instellenRijweg_1 with id GH1_15937247979422 ended up in an unexpected Rijweg toestand
- last status of request instellenRijweg_1 with id GH1_15937247979422 is Instelopdracht_afgekeurd

The ARC for this entry was at time 2020-07-02T21:19:57.9752Z, with action instellenRijweg_1 and opdrachtIdentifiatie GH1_15937247979422 with Seinbediening: ROZ .

IDCR entries for the status of the route

Timestamp	OpdrachtID	Attributes	ToestandRijweg
21:19:58.0694	GH1_15937247979422	RR IR	Rust
21:19:58.0857	GH1_15937247979422	IR	Gereserveerd
21:19:58.0998	GH1_15937247979422	IR	Voorbereid
21:19:58.1400	GH1_15937247979422		Instelopdracht_verstuurd
21:19:58.1609	GH1_15937247979422		Instelopdracht_afgekeurd

IDCR entries for the status of the request

Timestamp	OpdrachtID	OpdrachtToestand	Acceptatie
21:19:57.9802	GH1_15937247979422	Controleren	
21:19:58.0686	GH1_15937247979422	Uitvoeren	
21:19:58.1616	GH1_15937247979422	Gereed	NietGeaccepteerd

Elements in, by or connected to this route

We will now go in detail into the state transition of the elements that are part of the current route of interest. The timelines for each element show IDCE information according to Beheercli standards where changed information is highlighted with the color orange. Bevnl log entries are shown in grey on the same timelines, attributes that change between entries are only shown.

Elements in this route

Entries for WISSEL-105B-ZL-PPLG-1

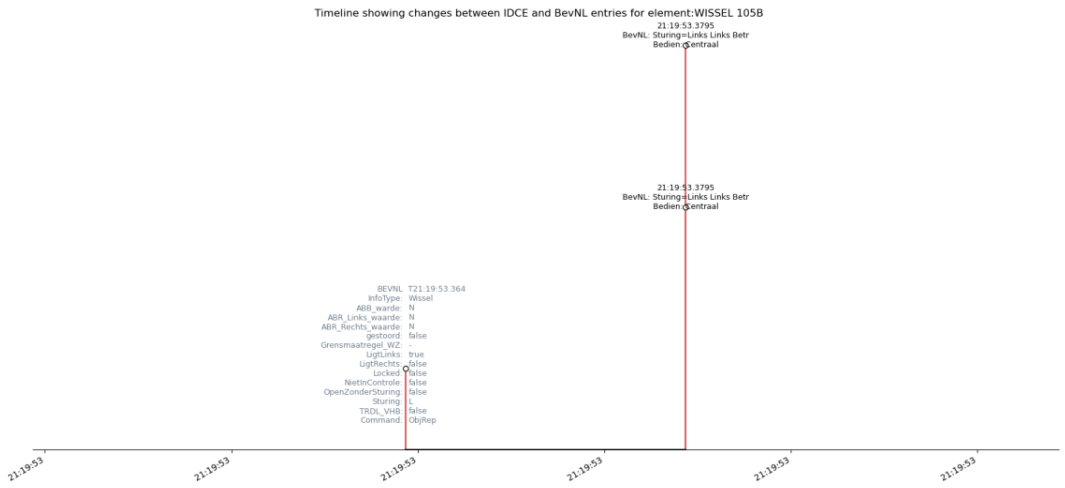
The element is only part of this active route process during the timeframe

Overview of IDCE entries for the element

Timestamp	OpdrachtID	Rijweg	ReqStand	Gebruikt	StandLRT	SturingLR	LigtL	LigtR	Locked	Betrouwbaar	NietInControle	OpenZonderSturing
21:19:53.3795				false		L	true	false	false	true	false	false

Overview of BevNL entries for the element

Timestamp	ABR_Links_waarde	ABR_Rechts_waarde	Betrouwbaar	LigtLinks	LigtRechts	Sturing	Locked	NietInControle	OpenZonderSturing	TRDL_VHB	Stand	Type of entry	Reply
21:19:53.364Z	N	N		true	false	L	false	false	false	false		Obj Rep	



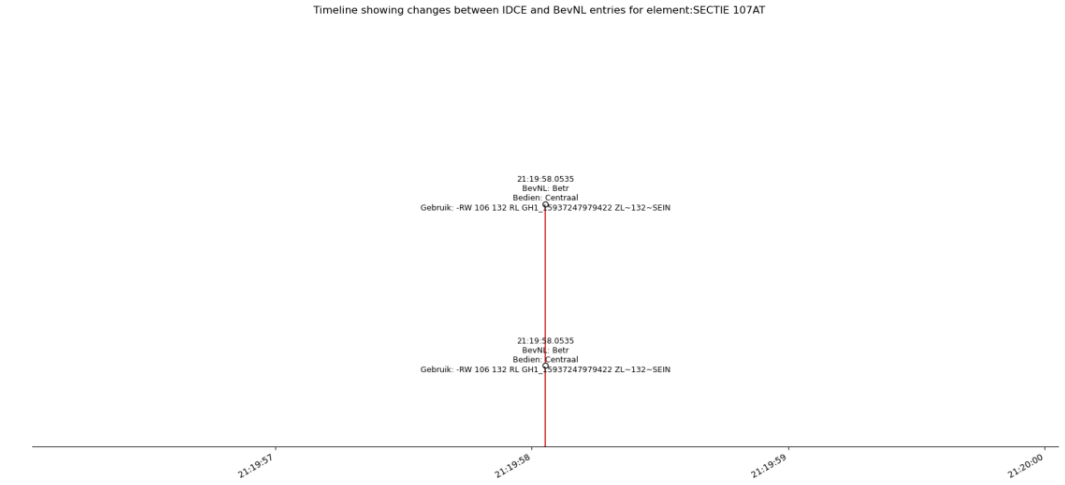
Entries for SECTIE-107AT-ZL-PPLG-1

The element is only part of this active route process during the timeframe

Overview of IDCE entries for the element

Timestamp	OpdrachtID	Rijweg	Gebruikt	Bezet	LigtInRijweg	RijwegRv.	GebiedRv.	Betrouwbaar
21:19:58.0535	GH1_15937247979422	106-132	false	false	false	true	false	true

There are no BEVNL entries for this element



Elements coupled to this element in this route

Entries for WISSEL-107B-ZL-PPLG-1

The element is only part of this active route process during the timeframe

Overview of IDCE entries for the element

Timestamp	OpdrachtID	Rijweg	ReqStand	Gebruikt	StandLRT	SturingLR	LigtL	LigtR	Locked	Betrouwbaar	NietInControle	OpenZonderSturing
21:19:53.3749				false		R	false	true	false	true	false	false

Overview of BevNL entries for the element

Timestamp	ABR_Links_waarde	ABR_Rechts_waarde	Betrouwbaar	LigtLinks	LigtRechts	Sturing	Locked	NietInControle	OpenZonderSturing	TRDL_VHB	Stand	Type of entry	Reply
21:19:53.363Z	N	N		false	true	R	false	false	false	false		ObjRep	

Elements in this route that are part of NPV situations

Some of these elements have already been handled for this route. They are:
WISSEL-107A-ZL-PPLG-1

Example RailText Report

The timeframe of interest for this report is from: 2020-07-02T21:19:39Z to: 2020-07-02T21:20:45Z. In this report we will look into active processes during this time frame and report events that are unexpected or could cause trouble.

In this report a red color will be used to highlight anomalous behaviour and yellow color will be to highlight changes between entries.

The following alerts (meldingen) were given within the timeframe.

Timestamp	Melding
21:19:58Z	Opdracht afgekeurd vanuit beveiliging.()

4 BevNL messages were sent between Astris and BevNL during the time period, out of which 2 requests and 2 responses, 1 requests were accepted and 1 declined.

There where 4 ARC entries within the timeframe of interest of which 2 route requests and 2 route related responses, 0 element requests and 0 element related responses.
The 2 requests that were accepted are:

- 1 instellenRijweg_1 for route 106-132RL,
- 1 instellenRijweg_1 for route 154-108LLRR.

Route request processes

We will now look into all routes that have active processes during the timeframe. The requests are grouped by routes. For each route we first go through an overview of the setting an behaviour of the route. Next we look into behaviour of IDCR entries and end with a detailed overview of element attributes.

The following routes had all requests normal and finished within the timeframe:
Route 154-ZL_108-ZL_LLRR
There is nothing alarming regarding these processes so we will not look more into them.

The following routes had some unexpected behaviour:
Route 106-ZL_132-ZL_RL
We will look more closely into them in the next section.

Route : 106-ZL_132-ZL_RL

The behaviour of this route is considered Incorrect for the reasons:

- command with ID GH1_15937247979422 got the value 'NietGeaccepteerd' from Astris in an IDCR entry,
- Rijweg BevNL command InstellenRoute was rejected with reason 'Opdracht afgekeurd vanuit beveiliging',
- request instellenRijweg_1 with id GH1_15937247979422 ended up in an unexpected Rijweg toestand and
- last status of request instellenRijweg_1 with id GH1_15937247979422 is Instelopdracht_afgekeurd.

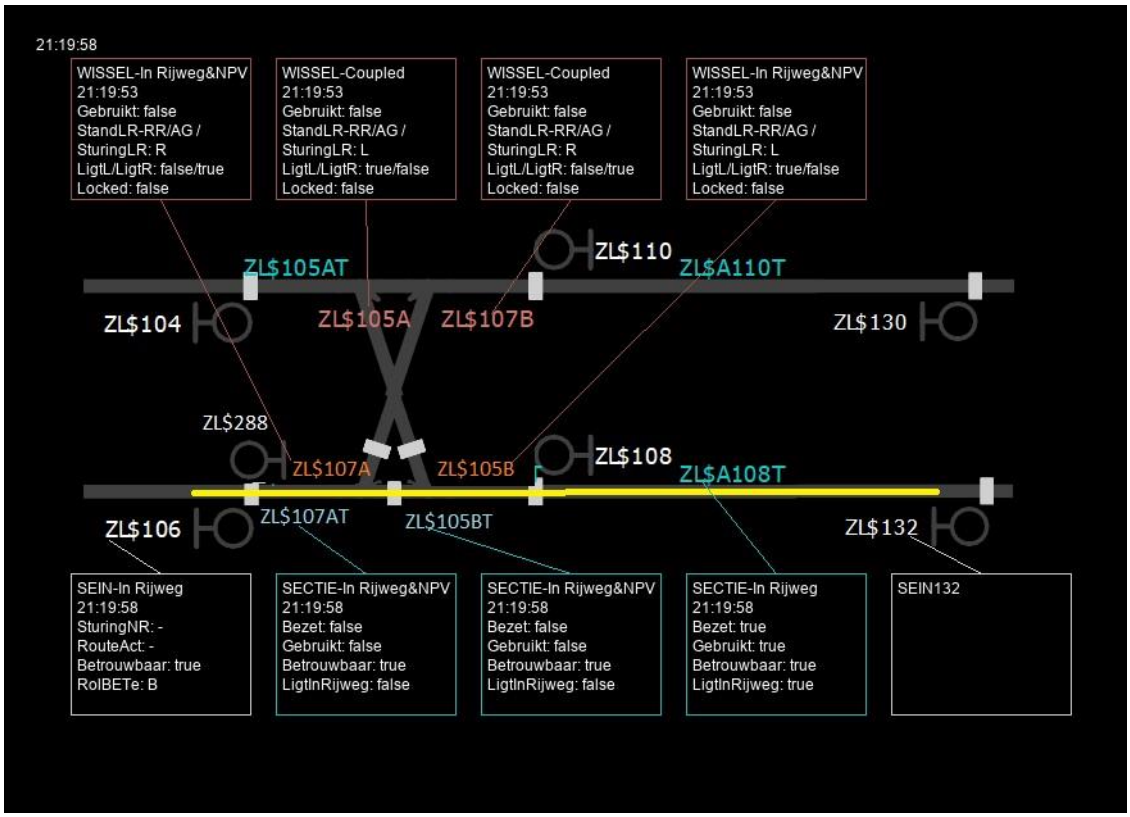
There is one element in this route that is also part of another route. It is the element SECTIE A108T.

Configuration setup for this route is - NormaalToegestaan: Ja, AutomaatToegestaan: Nee, ROZToegestaan: Ja, STSToegestaan: Ja, Geelectrificeerd: Ja, AstrisMoetVoorbereiden: Ja, ElementGestuurdeBeveiliging: Nee. Furthermore the VoorSectieIdentificatie of the route is: sectie A86T.

Railway overview for this route

The railway overview show the layout of the raiway with the elements. Each railway overview shows the most recent element attributes at a specific point in time. The route of interest is marked with a yellow line.

State of all elements in the route at time: 21:19:58



BevNL entries for this Rijweg

1 requests got rejected by the BEVNL layer. The requests that got rejected are InstellenRoute with reason 'Opdracht afgekeurd vanuit beveiliging'.

Timestamp	Command	CorrelatieId	BeginSein	EindSein	Acceptatie	AfkeurRedenNr	AfkeurRedenTekst
21:19:58.1284	InstellenRoute	5	SEIN-106-ZLPPLG-1	SEIN-132-ZLPPLG-1			
21:19:58.139Z	Antwoord	5	----	----	N	1	Opdracht afgekeurd vanuit beveiliging.

Requests for this route

Requests instellenRijweg_1

Some behaviour was considered unexpected for this request, that is because of:

- command with ID GH1_15937247979422 got the value 'NietGeaccepteerd' from Astris in an IDCR entry
- request instellenRijweg_1 with id GH1_15937247979422 ended up in an unexpected Rijweg toestand
- last status of request instellenRijweg_1 with id GH1_15937247979422 is Instelopdracht_afgekeurd

The ARC for this entry was at time 2020-07-02T21:19:57.9752Z, with action instellenRijweg_1 and opdrachtIdentifiatie GH1_15937247979422 with Seinbediening: ROZ .

IDCR entries for the status of the route

Timestamp	OpdrachtID	Attributes	ToestandRijweg
21:19:58.0694	GH1_15937247979422	RR IR	Rust
21:19:58.0857	GH1_15937247979422	IR	Gereserveerd
21:19:58.0998	GH1_15937247979422	IR	Voorbereid
21:19:58.1400	GH1_15937247979422		Instelopdracht_verstuurd
21:19:58.1609	GH1_15937247979422		Instelopdracht_afgekeurd

IDCR entries for the status of the request

Timestamp	OpdrachtID	OpdrachtToestand	Acceptatie
21:19:57.9802	GH1_15937247979422	Controleren	
21:19:58.0686	GH1_15937247979422	Uitvoeren	

21:19:58.1616	GH1_15937247979422	Gereed	NietGeaccepteerd

Elements in, by or connected to this route

We will now go in detail into the state transition of the elements that are part of the current route of interest.

Elements in this route

Entries for WISSEL-105B-ZL-PPLG-1

The element is only part of this active route process during the timeframe

Overview of IDCE entries for the element

Timestamp	OpdrachtID	Rijweg	ReqStand	Gebruikt	StandLRT	SturingLR	LigtL	LigtR	Locked	Betrouwbaar	NietInControle	OpenZonderSturing
21:19:53.3795				false		L	true	false	false	true	false	false

Overview of BevNL entries for the element

Timestamp	ABR_Links_waarde	ABR_Rechts_waarde	Betrouwbaar	LigtLinks	LigtRechts	Sturing	Locked	NietInControle	OpenZonderSturing	TRDL_VHB	Stand	Type of entry	Reply
21:19:53.364Z	N	N		true	false	L	false	false	false	false		Obj Rep	

Entries for SECTIE-107AT-ZL-PPLG-1

The element is only part of this active route process during the timeframe

Overview of IDCE entries for the element

Timestamp	OpdrachtID	Rijweg	Gebruikt	Bezet	LigtInRijweg	RijwegRv.	GebiedRv.	Betrouwbaar
21:19:58.0535	GH1_15937247979422	106-132	false	false	false	true	false	true

There are no BEVNL entries for this element

Elements coupled to this element in this route

Entries for WISSEL-107B-ZL-PPLG-1

The element is only part of this active route process during the timeframe

Overview of IDCE entries for the element

Timestamp	OpdrachtID	Rijweg	ReqStand	Gebruikt	StandLRT	SturingLR	LigtL	LigtR	Locked	Betrouwbaar	NietInControle	OpenZonderSturing
21:19:53.3749				false		R	false	true	false	true	false	false

Overview of BevNL entries for the element

Timestamp	ABR_Links_waarde	ABR_Rechts_waarde	Betrouwbaar	LigtLinks	LigtRechts	Sturing	Locked	NietInControle	OpenZonderSturing	TRDL_VHB	Stand	Type of entry	Reply
21:19:53.363Z	N	N		false	true	R	false	false	false	false		Obj Rep	

Elements in this route that are part of NPV situations

Some of these elements have already been handled for this route. They are:

WISSEL-107A-ZL-PPLG-1