Master Thesis

UNIVERSITY OF TWENTE.

Engineering Technology

Mechanical Engineering - Maintenance and Operations

Mitigating software failures impacts into IT and OT system design through RAMS

September 10th, 2024

Candidate:

Dimitar Lishev

s2883112

DPM 2143

Principal company:

Nederlandse Spoorwegen

Treinmodernisering

Supervisors:

Ir. Arno T. Kok
PHD candidate
University of Twente

Ir. Ferry Verrijzer
Senior Engineer
Nederlandse Spoorwegen

Dr.Ir. Willem Haanstra
Assistant Professor
University of Twente

Preface

Historically, the railway industry has been mechanically driven, but nowadays, with increasing digitization, Information Technology (IT) is gaining importance in this Operational Technology (OT) driven industry. This digitization brings several challenges for the Dutch Railways (NS) as an asset management company. For instance, multidisciplinary teams have been set up to integrate IT and OT systems within the rolling stock. Such teams traditionally use RAMS analysis to assess the impact of a certain change on the overall performance and maintainability of the system. This helps in the identification of possible failures that can occur in the system and the effects that these failures can have. However, within such analyses, the failures that arise from IT systems and specifically the software malfunctions are often overlooked. This is concerning, as in today's systems, software can be responsible for providing up to 50% of the total system functionality Carlson [2012]. This raises a need for research to be done on the possible methods and techniques for how software failures can be assessed, modelled and mitigated within the railway industry, which will bring the availability of systems that heavily rely on software significantly higher.

Abstract

In today's digitally driven world, the convergence of Information Technology (IT) and Operational Technology (OT) systems within the railway industry presents significant opportunities and challenges. The emerging importance of IT into the operations of rolling stock puts the software systems high into the functional criticality scale. This master thesis, conducted in collaboration with Nederlandse Spoorwegen (Dutch Railways), focuses on mitigating the impacts of software failures in IT and OT system design through the application of Reliability, Availability, Maintainability, and Safety (RAMS) methodologies. The research identifies the need for enhanced strategies to prevent software failures, which are increasingly critical in modern railway operations.

Employing a combination of qualitative and quantitative methods, the study first explores the importance and understanding of Information Technology (IT) and Operational Technology (OT) in the railway industry to set a new point of view for dealing with these systems from a maintenance perspective. Afterwards, the current RAMS practices applied to IT and software systems were explored, integrating insights from literature and interviews with engineers from the Dutch Railways. The core of the research involves adapting Reliability-Centered Maintenance (RCM) methodologies, traditionally applied to hardware, to address software failures. Various assistive tools are introduced to the RCM method, such as Software Functional Analysis, together with Software Failure Mode and Effect Analysis (SFMEA) and RCM logic tree to create a concise model which can serve as an effective tool to model and mitigate failures originating from the software within the IT and OT systems. Moreover, an Excel-based tool was developed to facilitate the implementation of the new process called Software Reliability-Centered Maintenance (SRCM).

The findings highlight the potential for improved system reliability and maintenance efficiency by incorporating software failure considerations into RAMS processes. The developed tool, validated through feedback from industry professionals, demonstrates practical applicability and offers a structured method for proactive software failure mitigation. The thesis concludes with a discussion on the implications for the railway sector and recommendations for future research to further enhance the reliability of IT-OT integrated systems.

Contents

Chapter 1

Introduction

In today's digitally driven world, Information Technology (IT) systems play a critical role in almost every aspect of human activity. From personal devices and corporate infrastructures to complex industrial processes and essential public services, the reliability of these systems has become paramount. Moreover, IT is becoming increasingly interconnected with operational technology (OT), which is responsible for the safe and reliable operation of nearly all machinery in the modern world. These systems depend significantly on software to ensure their reliable and predictable operation, making it crucial to address potential software failures as a key aspect of their maintenance. However, despite notable progress in the field of software engineering, software failures continue to be a persistent and expensive problem Sillito and Kutomi [2020]. Such failures can result in significant consequences, such as revenue losses, disruptions in operations, and potentially threatening human lives.

Two examples of critical software failures can be given from the aerospace industry. Firstly, on October 7, 2008, Qantas Flight 72, an Airbus A330, made an emergency landing following an in-flight accident that included a pair of sudden, uncommanded pitch-down manoeuvres. The Australian Transport Safety Bureau (ATSB) investigation found a fault with one of the aircraft's three Air Data Inertial Reference Units (ADIRUs) and a previously unknown software design limitation of the Airbus A330's fly-by-wire flight control primary computer (FCPC) ATSB [WA, 7 October 2008, VH-QPA Airbus A330-303]. It was observed that there were two inputs for the software that were interchanged and thus the software put the plane into a deep nosedive, endangering everyone onboard. The second example is when the Boeing 737 MAX faced a significant crisis due to two crashes that occurred in October 2018 and March 2019. The crashes were linked to the aircraft's Maneuvering Characteristics Augmentation System (MCAS), a flight control software system. The MCAS was designed to prevent stalls, but it was found to have design shortcuts that made the new plane seem like an old, familiar one. This led to a situation where the aircraft would take a nose-dive shortly after takeoff. It is observed that a hardware flaw was tried to be hidden by software in the design phase of the aircraft, but it was not rightly implemented. After the crashes, Boeing installed software updates and issued further training to pilots Travis [2024].

1.1 Challenges and gaps

1.1.1 Developments in the Railway Sector

In the railway industry, train operations have undergone a significant digital transformation, driven by the integration of Operational Technology (OT) and Information Technology (IT) Poliński and Ochociński [2020]. This transformation has led to the development and deployment of various systems designed to enhance

train movement, safety, and the overall passenger experience. Nederlandse Spoorwegen (NS), also known as The Dutch Railways, is increasingly implementing IT solutions within its trains and infrastructure, a trend that aligns with broader developments in the railway sector.

The integration of IT and OT has become central to modern railway operations. The European Rail Traffic Management System (ERTMS) is a prime example of this evolution, where advanced IT systems are integrated with traditional OT systems to improve efficiency and safety. However, the convergence of these technologies introduces new complexities and challenges. The interdependence of IT and OT systems means that failures in one domain can have cascading effects on the other, potentially leading to operational disruptions and safety risks. The challenge lies in managing this integration effectively, ensuring that both IT and OT systems are robust, interoperable, and resilient to failures. This topic will be further discussed in Chapter 3.

1.1.2 Role of Software

Software plays a crucial role in the modern railway environment, controlling everything from train scheduling and signalling to passenger information systems and predictive maintenance. However, with the increasing reliance on software, the potential for software-related failures also rises. These failures can stem from bugs, design flaws, or integration issues between IT and OT systems. The complexity of railway software systems, often involving multiple layers of legacy and modern code, makes it challenging to predict and prevent failures effectively. Current methodologies, such as Reliability, Availability, Maintainability, and Safety (RAMS), while useful, are often inadequate for addressing the specific risks associated with software failures in railway systems Kok et al. [2023]. This inadequacy creates a significant gap in the industry's ability to manage software-related risks effectively.

1.1.3 Role of Design

Design plays a pivotal role in the development of resilient railway systems, particularly in the context of IT and OT integration. The design phase is critical for identifying potential failure modes and mitigating risks before they manifest in operational environments. However, traditional design methodologies may not fully account for the complexities introduced by the integration of IT and OT systems. For example, the interaction between software components and physical systems can create unforeseen failure modes that are difficult to anticipate using conventional design approaches. This underscores the need for more advanced design methodologies that consider the unique challenges of IT/OT integration, including the dynamic interactions between software and hardware, as well as the evolving nature of cyber threats.

1.1.4 Failure Mode and Risk

The integration of IT and OT systems in the railway sector introduces new failure modes that were not traditionally considered in the purely mechanical systems of the past. These failures can originate from a variety of sources, including software bugs, communication breakdowns between IT and OT systems, and cybersecurity breaches. The risks associated with these failures are significant, potentially leading to service disruptions, safety hazards, and financial losses. Despite the critical nature of these risks, existing risk assessment frameworks are often not equipped to handle the complexities of IT/OT integration. For instance, while RAMS analyses provide a foundation for assessing system reliability and safety, they may not fully capture the software-related risks inherent in modern railway systems. This gap in risk assessment methodologies highlights the need for new approaches that can more accurately predict and mitigate the risks associated with IT/OT integration.

Given these challenges, the Dutch Railways (NS) is actively exploring new methodologies and technologies to enhance its ability to anticipate and avert malfunctions in its IT systems. This research has identified a critical opportunity to address these gaps by developing more robust approaches to managing the complexities of IT/OT integration in the railway sector. By doing so, NS aims to improve the reliability and safety of its operations, ensuring that software failures do not stand in the way of daily operations or lead to potential accidents.

1.2 Main goal

This master thesis focuses on the need for efficient strategies for preventing software failures, as they are considered an important blind spot in the Dutch Railways assessment process, as will be shown further in this report. The objective is to investigate and assess different methods that can improve the availability and reliability of IT and OT systems, with a particular emphasis on the software component. This necessitates the evaluation of these failures through modifying already present methodologies Tafur et al. [2021] or completely evolving them to fully accommodate the present challenge of IT and software failures Hehenberger et al. [2016]. This lays the foundation of the proposed research goal, which aims to address the critical question:

How can the effects of software failures be modelled during the design of Information Technology (IT) and Operational Technology (OT) systems in the railway systems?

The research question aims to delve into the field of rolling stock maintenance by researching how to model software failures and mitigate them with suitable maintenance strategies, similar to how it is done for hardware failures. For this, a comprehensive literature review was carried out to find what are the current effective methods for mitigating software failure, presented in Chapter 4. These strategies include analysing potential failures during the design phase and implementing preventive controls during the operational phase. By identifying potential software failure points and their impacts on IT and OT systems early on in their design phase, this approach seeks to enhance the reliability and effectiveness of rolling stock systems, optimise maintenance schedules, and reduce costs for the maintenance of software systems. Adopting innovative strategies, as discussed in this report, allows the field of rolling stock maintenance to adapt to technological advancements.

1.3 Research objectives and questions

Considering the aforementioned challenges and to address the main research goal, the following objectives will be established for this research:

1. Identify the main way of understanding the nature and failure mechanisms for IT systems and the methods of prediction of their key software failures in the rolling stock;
2. Develop a method to incorporate the effects of software failures into IT-OT converged systems within the RAMS process;
3. Demonstrate the added value of incorporating the effects of software failures concerning system performance.

By achieving these objectives, this research will try to contribute to the improvement of RAMS processes in the principal company and also in the whole railway industry. Moreover, the results of this study could also have implications for other sectors that depend on Physical systems incorporating software elements.

1.4 Partial Conclusion

This thesis aims to enhance the comprehension of software failure mitigation and provide practical solutions to improve the resilience of software systems. By pursuing the set goal and objectives, it seeks to enhance the development of reliable and efficient software, thereby facilitating the continuous digital revolution in the railway industry and also in other industries influenced by the same revolution.

Chapter 2

Research outline

2.1 Research methodology

In this research, both qualitative and quantitative analysis were utilised to research and develop a methodology for modelling software failures within the Dutch Railways (NS). The qualitative analysis involved interviewing engineers to gather insights on common causes of software failures and opinions on the current prevention controls. The quantitative analysis included collecting data on past software failures and using them as potential input for the suggested methods to identify the possible strategies that can be proposed to help mitigate future failures based on severity and frequency. By combining these two approaches, a comprehensive methodology was developed that can be used to model and mitigate software failures in the future.

The research highlights the challenges of dealing with the differences between informational and operational technologies and aims to improve the understanding of these technologies and their failures, aiming to bridge the gap between IT professionals and railway engineers and strengthen their communication and collaboration. This thesis then takes a turn on the software component of both technologies as the blindspot of the current assessment processes. It suggests a method for detailed examination of software failures and mitigation of such failures, which is based on the methodologies used in other industries and in the academic world. These findings aim to contribute to a more predictable and reliable performance of the software in the railway sector. Figure 2.1 outlines the research steps taken to achieve these objectives.

2.2 Literature review

First, a literature review about different understandings of IT and OT technologies across industries was executed. This was done to compare them to current understandings within NS and bridge the gap if such is found to exist. Then a comprehensive examination was carried out of the existing techniques for evaluating software failures that are documented in the literature. This is done to establish a strong knowledge base regarding the methodology employed to minimise the consequences of software failures. All the analysed methodologies and tools are specifically designed to improve the reliability of software.

2.3 Information gathering phase

2.3.1 Research of current RAMS methods

Initially, the research focused on examining the present-day RAMS methodologies employed in NS to assess IT and OT systems, encompassing both their hardware and software components. This knowledge serves as the basis for the current assessment of software components and failures, as well as identifying the main areas where gaps exist within the currently applied procedures. Additionally, the proposed methodology has taken into account previously used strategies to make sure it is not entirely unique and unfamiliar, when it is similar to already used ones within the organisation, it encourages future implementation within the company.

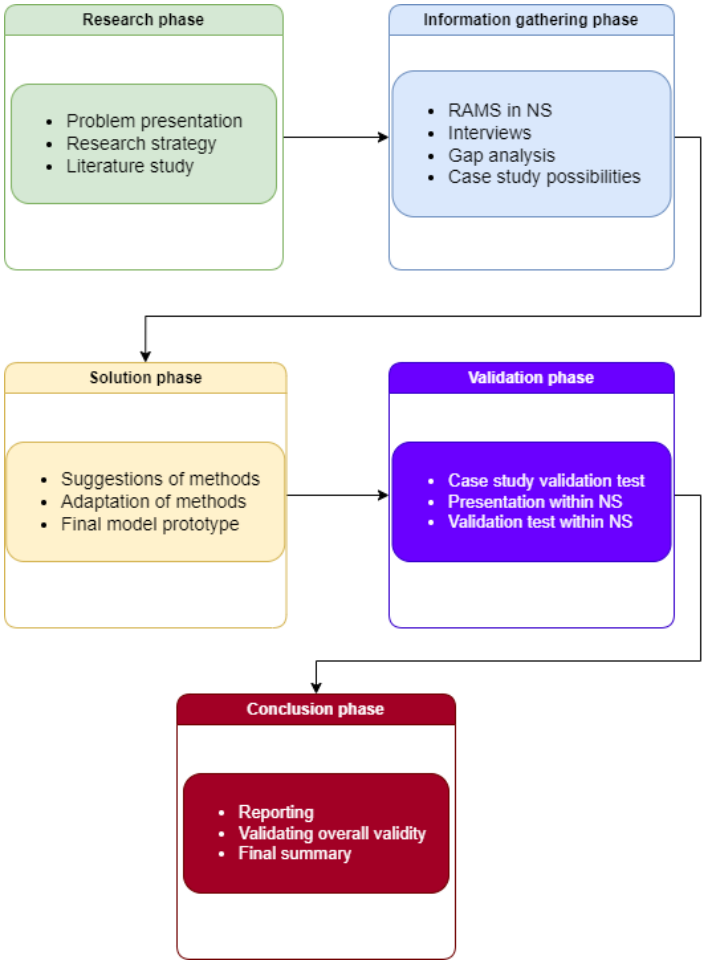

Figure 2.1: Research process map

Upon completing the gathering of information and analysis of the existing methods, the subsequent task involved identifying potential areas of enhancement in the current assessment process. By identifying the primary weaknesses in the assessment of both IT and software failures, the proposed methodology was able to specifically target and resolve these issues. Furthermore, by harmonising the proposed approach with the existing methodologies, it guarantees seamless integration and execution within the organisation, resulting in a proposed process that ensures smooth integration alongside existing ones, which will increase the chance of bringing value to the company if applied.

2.3.2 Interview phase

Fifteen engineers from various departments were interviewed to gather information regarding their understanding of IT, OT, and software failures. This step was crucial in assessing the current perceptions of mechanical engineers regarding the recent digitization of trains, as well as their views on the severity and underlying causes of software failures. Additionally, their opinions and suggestions were also used in developing the potential solution for a software failure assessment tool. By adopting this approach, it was guaranteed that the outcome of this thesis would yield not only academic but also organisational benefits.

2.4 Solution phase

The acquired knowledge from the literature review and the information-gathering phase was utilised to develop a new approach for modelling software failure. The primary objective was to adapt the existing RAMS methodology to effectively address potential software failures and fill the gap both academically and in the industry for such a methodology. The Reliability-Centered maintenance method was adapted so failures coming from software can fit into the already used framework that this method presents. Then, an Excel tool was developed and demonstrated to the company to illustrate how these ideas can be implemented in the

industry, particularly in the case of the Dutch Railways, through the application of a specific case study.

2.5 Validation and Evaluation phase

Subsequently, a video was made to demonstrate the step-by-step procedures for utilising the Excel model, and it was presented to the interviewed engineers who were already familiar with this research. In addition, participants were given a questionnaire to complete in order to provide their assessment of the suggested concepts and model. Overall, the evaluation process allowed for a comprehensive analysis of the effectiveness of the Excel model in addressing the research objectives. The results of the evaluation will be discussed in detail in the results and evaluation section.

Chapter 3

Operational versus Informational Technology

3.1 Introduction

This chapter begins with an explanation of what are Operational and Informational technologies. Furthermore, it demonstrates the current and future differences and convergence of these two technologies together with the Physical system and highlights the significant role that software will have. Next, the information gathered via the conducted interviews within the Dutch Railways (NS) is presented and a comparison is drawn with the literature, to formulate a suitable and precise definition from a maintenance perspective. One of the objectives is to highlight the significance of the roles that IT will have in the future and facilitate a clearer understanding of the definition of this technology. The definition and understanding of what is a software failure is also put on the table, as again, a comparison is carried out between the literature and current opinions within NS. It is important to clarify the understanding of what is software failure before setting the way for its assessment.

3.2 Main understandings in Literature

3.2.1 Operational Technology (OT)

Operational Technology (OT) refers to the hardware and software systems that are used to monitor and control physical devices, processes, and events in an industrial environment Gartner [2021]. OT systems are essential for managing infrastructure operations, ensuring safety, and optimising performance in real-time. These systems are commonly found in industries such as manufacturing, transportation, energy, and utilities. OT encompasses a range of technologies, including industrial control systems (ICS), supervisory control and data acquisition (SCADA) systems, and programmable logic controllers (PLCs) Reussner et al. [2019]. The primary reason for failure in OT systems is typically attributed to hardware components, which are prone to failure as a result of faulty design, extensive wear, or inadequate maintenance. Nevertheless, the software component can also be the cause of failures of these systems, primary software failures are caused by the wrong initial design of the software. During the operational phase, software in operational technology (OT) systems tends to exhibit more constant inputs and thus higher reliability compared to software in information technology (IT) systems, which usually experience less predictable inputs Stouffer et al. [2023]. The primary focus of OT is on the physical operations and the direct control of machinery and processes, ensuring safe and reliable operations of everyday tasks.

3.2.2 Informational Technology (IT)

Information Technology (IT), on the other hand, involves the use of computers, storage, networking devices, and other physical devices, infrastructure, and procedures to create, process, store, secure, and exchange all forms of electronic data Dewett and Jones [2001]. Like OT systems, IT systems also consist of hardware and software components. However, in IT systems, the software component is more susceptible to failures due to its increased complexity and greater number of inputs. The design phase is often the stage where software failures are most likely to occur, typically due to inadequate initial software design and incomplete requirements. IT systems are designed to support complex operations and decision-making through data processing, information storage, and communication technologies. Examples of such systems include

enterprise resource planning (ERP), and customer relationship management (CRM). In the railway industry, such systems include the Closed-Circuit Television System (CCTV) and the European Rail Traffic Management System (ERTMS), which are examples of both non-safety critical and safety-critical systems, respectively, that closely rely on transferring information in real-time, thus heavily rely on software Towhidnejad et al. [2003]. Overall, IT systems focus on data management, networking, and application development to enhance operational efficiency, user experience, and productivity Jorgenson and Stiroh [1999].

3.3 Differences between OT, IT and Physical systems

While OT and IT systems are integral to modern industrial and business operations, they differ fundamentally in their focus, design, and operational goals Jorgenson and Stiroh [1999]. OT is concerned with the control and automation of physical processes, prioritising safety, reliability, and real-time performance. IT, conversely, is centered on managing information and enhancing processes, emphasising data integrity, cybersecurity, and operational efficiency. OT systems are typically designed to operate in real-time and are often isolated to prevent external access, whereas IT systems are more interconnected and accessible for remote management and collaboration, which makes them more vulnerable to design flaws and cyber-attacks. The convergence of OT and IT is becoming increasingly important as industries seek to leverage data from both domains to improve overall operational efficiency and decision-making capabilities.

One must consider the distinctions between Information Technology (IT), Operational Technology (OT), and the Physical system (PS), shown in Figure 3.1. Typically, the latter exhibits an analogy to the hardware components forming the machine or overall framework, for example, the door mechanism of the train is considered the Physical system, the PLC that controls it is the OT and the connection between the PLC and the main computer of the train or a maintenance device can be considered as IT. Previously, these systems were primarily operated by mechanically driven components, but nowadays, they are mainly controlled by OT systems like PLCs. This has facilitated the implementation of safer and more dependable management of the physical systems. Presently, Information Technology has further enhanced the ability to exercise superior control and monitoring over both Operational Technology and Physical Systems.

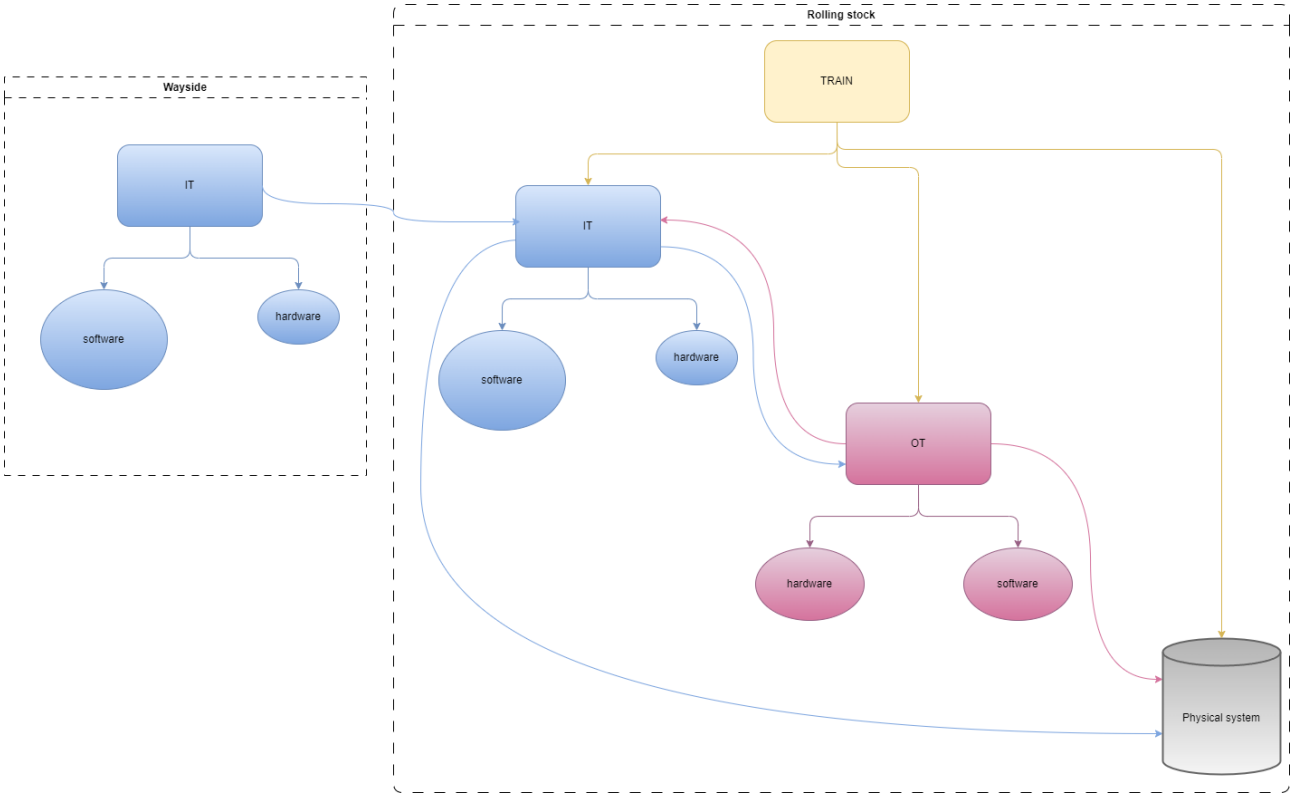

Figure 3.1: The connection between IT, OT and PS in the Railway industry

3.4 IT and OT convergence

The convergence of Information Technology (IT) and Operational Technology (OT) is progressively transforming the industry. With the development of digital technologies, the distinction between these two technologies is becoming less apparent, resulting in a more unified and effective operational environment. Such systems are also known as Cyber-Physical systems (CPS) Monostori [2014], where "Cyber" typically refers to Informational and "Physical" to Operational Technology and the mechanical components of the system Kok et al. [2023].

Similarly, the railway industry is undergoing a transformation as IT is being incorporated Kraeling and Fletcher [2017], leading to significant changes in the operation and passenger experience of train journeys. The convergence between IT, OT and PS is outlined in Figure3.1, which shows the connection between the two technologies and the Physical System within the train and the connection between the rolling stock and the surrounding infrastructure. This infrastructure includes systems for controlling and monitoring such as signalling, power distribution, and train operations Murray et al. [2017]. It will be extensively implemented in the coming years and will be crucial for the safe and efficient operation of the train, transforming it more and more from informational to operational technology. That will bring many changes as this field has long relied on robust OT systems, often using proprietary protocols and hardware, which were typically isolated from IT networks to maintain security and reliability. However, the rise of digital transformation initiatives has necessitated greater integration between IT and OT to enhance operational efficiency, safety, and reliability.

The stated above means that software plays a pivotal role in the convergence of IT and OT, driving the integration and functionality of complex railway systems. According to Kamal et al. [2016], IT serves as the foundation for data transformation and integration, supporting various OT applications like real-time measurement, control systems, and safety systems. The software facilitates communication between IT and OT systems, enabling seamless data flow and operational control. Consequently, if the software fails, it will result in a disturbance in the seamless flow of data and control in both IT system operations and the communication between IT and OT. This necessitates the development of appropriate strategies during both the design and operational stages to minimise the potential failures that may arise from inadequately designed or maintained software.

3.5 IT and OT within NS

Providing an exact definition for Information technology is a complex and debatable topic in the industry. To investigate the understanding of Operational and Informational technologies within NS, during the interview phase, the engineers were asked to provide their definitions of these two technologies. It is crucial to have a clear understanding of emerging technologies, especially in the field of IT. People who are not regularly involved with these systems often misunderstand and underestimate their main functions. This can lead to ignorance in assessing potential failures and, more importantly, underestimating the effect of these failures. This can be particularly dangerous when there is complete convergence between IT and OT. This paragraph aims to present the current understanding of both IT and OT within the principal company and align them with the literature mentioned above. The goal is to develop the most suitable definition for this research and potentially for the company in the future.

Fifteen engineers from different backgrounds were interviewed, the exact positions are presented in Chapter 7. The main topics of conversation were their understanding of IT, OT and software within the rolling stock components and the current method for assessing failures that originate from software. Also, some experiences from such failures were discussed. Two of the questions that all of them were asked were: "What is IT?" and "What is OT?", in their opinion and understanding. The majority provided explicit responses, although there were a few who did not clearly articulate their perspective. After analysing all the responses, the identical answers were merged, resulting in the formation of eight distinct opinions/ shown in Table 3.1. The Table also displays additional insights regarding working with OT and IT that were provided during the interviews.

Opinion	OT Understanding	IT Understanding	Additional Insights
1	Necessary to drive safely the train (traction system, braking system, TCMS, diagnostic system)	All of the other systems	-
2	OT is everything safety-critical in the train	IT is everything non-safety critical	IT failure can wait, OT failure cannot
3	Equipment needed to drive the train (bogies, train, relays, systems in trainsets)	Equipment that enhances the passenger experience	Main differences between IT and OT is maintenance and problem-solving
4	Hardware related	Software related	-
5	Systems ensuring train operation (ERTMS, traction, braking)	Passenger information systems (public Wi-Fi, portals, information screens)	TCMS considered IT due to Linux but essential for train, thus OT
6	Parts that are responsible for driving the train	Communication parts to the train (OBIS network)	IT delivers packages to OT, OT processes them
7	Train is purely OT	Wayside is IT, potential future operational relevance	Impact of failure is key, not the label (IT/OT)
8	Everything on the train	Everything off the train	Not universally agreed

Table 3.1: Definitions and Opinions on OT and IT from NS engineers

Definition of IT/OT	Description
Definition perspective 1	
Main understanding for OT	- Includes all hardware components responsible for driving the train systems, such as bogies, train equipment, relays, and systems for traction and braking.
Main understanding for IT	- Encompasses all communication systems and software-related components, including passenger information systems and network infrastructure.
Definition perspective 2	
OT vs IT: Hardware vs Software	- IT is perceived as software-related, while OT is hardware-related.
	- IT includes communication parts to the train, such as the OBIS network.
	- OT includes essential surfaces and systems for traction and braking, while IT covers passenger information systems and non-essential components.
	- If one component of OT fails then it is only one of it that is not available, but if IT system fails then there are all of the components associated with it that are not available anymore.
	- TCMS, although running on Linux, is considered essential for running the train, thus seen as both IT and OT.
Definition perspective 3	
IT/OT: Train vs Wayside Systems	- The train is typically viewed as OT, while wayside systems may lean towards IT, with the distinction sometimes blurred.
	- The distinction between IT and OT is not always critical; the focus is on the impact on operations when failures occur.

Table 3.2: Different perspectives of IT and OT definitions

It can be observed that the opinions of the engineers can be separated into three distinctive silos, shown in Table 3.2:

1. The first is the most common understanding, which is also the closest to the literature.
2. The second resembles that some of the engineers understand the two systems as one (OT) is purely hardware and the other (IT) being purely software.
3. The third one is where the train is observed as a purely OT system and therefore everything that is in the wayside - the infrastructure around the train is observed as IT.

The interviews with engineers revealed diverse perspectives on the definitions and distinctions between Information Technology (IT) and Operational Technology (OT). The general opinions stated:

- **OT** is perceived as encompassing the hardware and systems essential for the safe and efficient operation of trains, including traction systems, braking systems, and the Train Control and Management System (TCMS).
- **IT** is often associated with software and communication systems, such as passenger information systems and public Wi-Fi.

A common theme is the criticality of OT systems, where failures can immediately impact train operations and safety, whereas IT failures, though inconvenient, are less urgent. Some engineers highlighted the integration of both technologies, noting that systems like TCMS, which run on software platforms, straddle the line between IT and OT. The impact of failures, rather than strict categorizations, was emphasized as a crucial consideration. Furthermore, while one view simplistically divides OT as everything on the train and IT as everything off it, this was not a universally accepted distinction. These varied interpretations underscore the complex, overlapping nature of IT and OT within the rail industry, driven by both functional roles and the criticality of systems.

3.6 IT and OT defined for the railway sector

Upon evaluating the literature available and the opinions of multiple engineers, the most relevant definitions for the railway industry, and specifically for NS, can be determined as follows:

- **Operational Technology (OT)** - *Includes all the systems that are responsible for the operational functions of a passenger train. These systems should ensure that the train can efficiently and securely board passengers and transport them from one location to another.*
- **Information Technology (IT)** - *Encompasses all the systems that are responsible for the transfer of information. This includes facilitating communication between operational technology (OT) systems, between the train and the external infrastructure, and between the driver/systems and the passengers.*

These definitions provide a clear framework for classifying IT and OT for the railway systems, useful for distinguishing between two technologies when necessary. They are mainly derived from the available literature, but sometimes they bring unnecessary confusion in the industry as seen by the interviews. Therefore to try to mitigate this confusion a suggestion will be made to shift the thinking more towards the second perspective in Table 3.2 - viewing these systems as part of a unified IT-OT or Cyber-Physical system and just categorize the failure as hardware and software ones. **Differentiating systems by their software and hardware components could prove more effective than merely categorising them as IT or OT from a maintenance perspective.** Engineers can align their perspectives with this framework, which could eliminate the confusion of which system is which and will focus the attention on the failures. The view for the systems should be more like what is shown in Figure 3.2, rather than the one in Figure 3.1. This approach ensures a thorough assessment of failures, eliminates uncertainties about system classification, and bridges the gap between specialists. Consequently, it becomes easier to identify and differentiate failure sources, making them simpler to model and predict.

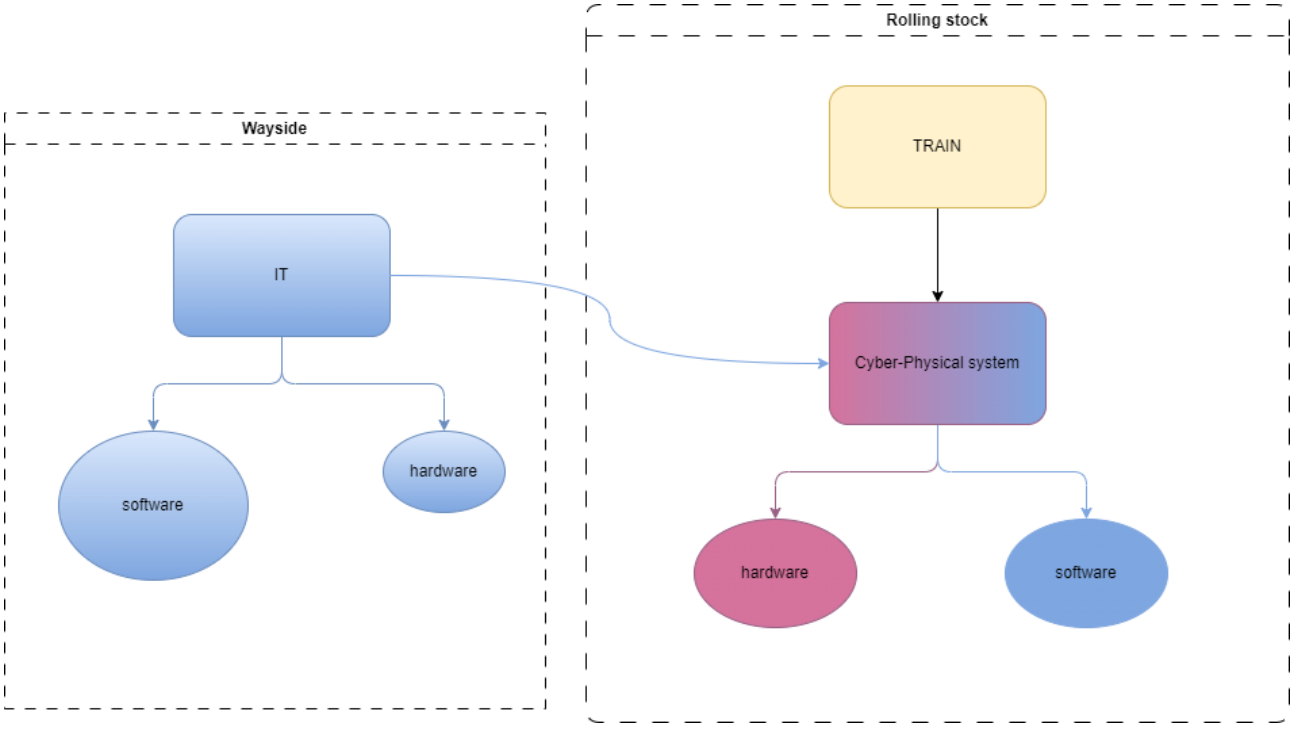

Figure 3.2: How all of the systems in a train should be observed from an operational perspective

3.7 What is Software failure?

As was proposed in the preceding section, to shift the perspective of existing systems just to systems that are composed of hardware and software, rather than as IT or OT systems, to simplify the identification of their failures. With this another issue arises, as hardware failures are well-known due to their nature and the experience of encountering them. However, software failures are occasionally misunderstood and therefore underestimated, especially by mechanical or electrical engineers. To enhance the clarification from this standpoint, a thorough literature review was conducted to define the concept of software failure.

All of the interviewed engineers were also asked about what they think a software failure is. However, not every interviewee could give a clear definition of it, as some also stated that such failures do not exist. After an analysis of the observed data, seven distinctive opinions could be highlighted, as shown in Table 3.3. This resulted in a rather interesting analysis, clearly showing again a range of diverse opinions and points of view, which could be harmful for the future mitigation of such failures. That is why the need for clarifying the definition of software failures arose.

Opinion #	Key Points
1	- Software issues often root in hardware problems. - Software does not inherently change or degrade.
2	- Sensor degradation affects inputs/outputs. - Software remains consistent; hardware aging is the issue.
3	- Failure occurs when design specifications and boundary conditions do not align.
4	- Software is systematic but can be affected by external changes. - Physical processes affecting software reliability are disputed.
5	- Software failure is defined by unmet requirements. - Software reliability should include detecting hardware problems and human errors.
6	- Hardware failure does not impact software reliability. - The design phase should include coping mechanisms for hardware issues.
7	- Software failures are due to programming errors, not inherent software flaws.

Table 3.3: Key Points on Software Failure from Interviewed Engineers

The opinions gathered from the interviewed engineers provide a diverse range of perspectives on the definition and nature of software failure, which can be compared to the established definition in the literature. The literature defines **software failure as a systematic issue caused by faults introduced during the development stages (requirements definition, design, and implementation).** According to this definition, once an error is fixed, it will not reoccur, and physical processes like stress and wear **do not** influence software reliability Manen et al. [2015]. The software excels at repetitive tasks without degradation over time but struggles with incidental tasks requiring adaptation or improvisation Jones [2000].

Comparing this to the engineers' opinions, several key points emerge. First, many engineers agree that software issues often root in hardware problems, emphasising that software does not inherently change or degrade (Opinions 1, 2, and 6). This aligns with the literature's view that physical processes do not impact software reliability. However, there is a recognition that external changes and environmental factors can affect software performance, which is somewhat acknowledged in the literature through the mention of unforeseen input combinations causing random failures (Opinions 4 and 5).

Furthermore, the engineers highlighted the importance of meeting design specifications and requirements, with failures often attributed to deviations from these (Opinions 3 and 5). This perspective underscores the systematic nature of software failure as defined in the literature. Interestingly, one engineer's view that software never fails but can be programmed incorrectly (Opinion 7) echoes the literature's point on faults introduced during the development process.

Laying on all of the above-stated research definitions, opinions and experiences, for this research, a specific definition of a software failure was derived to clear out the confusion about what exactly a software failure is. After taking into consideration all of the engineers' opinions and what is stated in the literature, the definition was specified as:

- **Software failure** - *The **inability** of the software to perform the intended functions and produce the desired output correctly according to the requirements and inputs. When the software failure is fixed, it will never show up again in the same manner. In specific situations, software reliability **can** be influenced by wear and tear in the hardware, and in some cases, hardware failure **could be** considered the root cause of a software failure.*

It should be said that this definition tries to alter the way of thinking about a software failure and bring the possible hardware failure as a root cause of software one. As the definition aligns with the literature until the point where it states that the deterioration of the hardware may impact the reliability of the software. This opinion is more compatible with the one deduced from the data obtained through interviews, and the examples given, both of which are primarily grounded in firsthand experience. Moreover, the author of this thesis tends to agree with this opinion more than the one presented in the literature. However, it is important to acknowledge that further research is needed to specifically investigate the underlying causes of software failures and determine whether hardware failures are a contributing factor and this thesis aims to spark the interest of such research.

3.8 Partial Conclusion

In conclusion, the presented content in this chapter shows a thorough examination of Operational Technology (OT) and Information Technology (IT), highlighting their merging and importance in the railway sector. The integration of these technologies into Cyber-Physical Systems (CPS) is revolutionising the industry.

The chapter suggests that IT and OT could be conceptualised as hardware and software components to make it easier to facilitate accurate assessment of their failures, based on insights from Dutch Railways (NS) engineers. Furthermore, it emphasises the importance of establishing a precise characterization of software failure in order to minimise the disparity between software and mechanical engineers.

All of this emphasized the change in the industry and the challenge presented by the recently emerged failures from new technologies, particularly from software which requires new approaches to be investigated and tested laying down the foundation for the next step of this research.

Chapter 4

RAMS for hardware and software

4.1 Introduction

RAMS (Reliability, Availability, Maintainability, and Safety) is an integrated discipline crucial for the operation and engineering of railway systems Mahboob and Zio [2018], aiming to achieve a safe, highly dependable, predictable, and available system without compromising safety and cost-effectiveness Muhammed Nor et al. [2021]. It is mainly applied to hardware components presenting numerous advantages and enhancing the overall reliability in the operations of different systems and components, especially in the operational phase Kumar et al. [2021], Zhang et al. [2021]. It is necessary to modify these methodologies to suit the emerging Informational Technologies Kok et al. [2023], particularly the software component.

As the software becomes increasingly responsible for direct operations of the Physical System, such as the train in the case of NS, it becomes more critical for safety. Also, NS is experiencing more and more software failures in its daily operations which lower the availability of the rolling stock and even could pose safety-related issues, Figure 4.1 shows the failures recorded in the ERTMS system for several years.

Figure 4.1: Failures recorded in the ERTMS system in recent years(figure blurred for public use)

Therefore, there is a need to ensure the safe and reliable functioning of these systems through methods like RAMS. Thus, this literature review was carried out to showcase the used RAMS methods for assessing software and step on them when researching the possible solution that NS is seeking for their challenges with software integration.

4.2 Hardware RAMS

RAMS includes numerous strategies for finding the fundamental cause of failures and making them predictable. One of them, Failure Mode, Effects and Criticality Analysis (FMECA) is an approach used to identify, analyze, and evaluate the potential risks associated with unexpected failure of rolling stock components in the railway industry Mahboob and Zio [2018]. It helps in assessing the performance of current maintenance practices and planning a cost-effective preventive maintenance program. This analysis helps organizations proactively identify weaknesses in their designs, processes, or products and implement corrective actions to mitigate these risks. By categorizing failure modes according to their criticality, FMECA enables engineers and decision-makers to focus their efforts on the most significant issues that could impact safety, performance, and reliability, thereby enhancing overall system robustness and dependability Carlson [2012].

Other methods include Fault Tree Analysis (FTA) Roberts and Haasl [1981], a type of failure analysis used in safety and reliability engineering to understand how systems can fail, identify ways to reduce risk and determine event rates of a safety accident or particular system-level level failures in the railway industry, FTA has been used to generate safety risk models for train collisions, evaluating the fundamental error and failure probability that could potentially lead to accidents, also commonly applied to all kind of critical infrastructure Márquez et al. [2016].

4.3 Software RAMS

The integration of software in safety-critical railway systems necessitates rigorous RAMS (Reliability, Availability, Maintainability, and Safety) evaluation to ensure the overall system's integrity, a task made challenging by the inherent complexity of software versus hardware components. Revision of key standards such as IEC 61508 Bell [2006], CENELEC EN 50128:2011 and also the updated CENELEC EN 50129:2018 reflects the industry's recognition of the need for enhanced software safety and reliability techniques, which are critical as software becomes a pivotal element in system performance. The paper by González-Arechavala et al. [2010] examines these evolving standards and methods, exploring their application not only in the railway sector but also in other high-stakes industries like nuclear and aeronautics, to foster advancements in system specification and thereby improve software reliability and safety. Moreover the standard CENELEC IEC 62628:2012 presents guidance on software aspects of dependability, highlighting some crucial aspects of the software component, also the newly emerged standard CENELEC EN 50716:2023 present the latest requirements for design and development of software for Railway Applications, recommending some viral insides for modelling the software which this report will refer to in the next chapter.

In the realm of software design and system safety, the application of Fault Tree Analysis (FTA) has proven to be a robust methodology. As demonstrated in the research by Towhidnejad et al. [2003], FTA, particularly in the form of Software Fault Tree Analysis (SFTA), can be integral during the requirements and design phases, offering a systematic approach to identifying safety-critical components and potential hazards. Another application for assessing software failure using FMEA and FTA is carried out by Medikonda and Ramaiah [2014]. The study presents a novel approach for enhancing software safety in safety-critical systems, specifically using a model Railroad Crossing Control System (RCCS) as a case study. The authors integrate Software Failure Modes and Effects Analysis (SFMEA), seen in Table 4.1, and Software Fault Tree Analysis (SFTA) to systematically identify and analyze potential software faults and their impacts. They demonstrate this method by applying it to RCCS, where they pinpoint critical software failure modes such as gate malfunction and improper train route changes, assessing the severity and devising preventative measures. The study emphasizes the differentiation between software and hardware failures and concludes that their integrated approach effectively identifies potentially hazardous software faults, thereby improving the safety and reliability of software-controlled systems. The SFMEA table highlights the significant focus on potential

software-related causes of failures. This is a crucial aspect that is currently absent in the analysis conducted within NS. It provides a convincing case for the necessary research and adaptation required.

Failure Mode	**Possible Causes**	**Effect**	**Severity of risk**	**Prevention And Compensation**
Gate not closed as train is passing through	a) sensor not detected by s/w	Train collision with passing road traffic leading to accidents	Critical	Software first checks the working status of gates each time the train is about to cross the gates
	b) gate motor mechanism is defective			
	c) s/w gives wrong command			
	d) s/w gives right command at wrong time			
Track change lever is not activated to change train route	a) sensor is not detected by s/w	Train fails to change its path from the outer track circuit to the inner track circuit leading to accident	Critical	Software first checks the working status of the track lever each time the train is about to enter the inner track loop
	b) track lever motor mechanism is defective			
	c) s/w gives wrong command to lever			
	d) s/w gives right command at wrong time			
	e) s/w fails to give a command to activate lever			
Control program software is corrupted	a) logic fault	Unpredictable sequence of operations leading to accident	Critical or Catastrophic	Algorithm logic is verified for accuracy. Data Structures and Memory overflow is checked.
	b) interface fault			
	c) data fault			
	d) calculation fault			
	e) memory fault			

Table 4.1: Failure Modes, Possible Causes, Effects, Severity of Risks, and Prevention and Compensation Methods Medikonda and Ramaiah [2014]

Another example on implementing FMECA applied in the railway sector that has a connection to IT failures is given by the study by Dinmohammadi et al. [2016], which introduces an in-depth examination of risk management for rolling stock in the railway sector, specifically through a case study of Class 380 train door systems in Scotland. Utilizing the Failure Mode, Effects, and Criticality Analysis (FMECA) approach, the paper aims to enhance maintenance strategies and improve the reliability and safety of railway operations by mitigating the risks associated with door system failures. According to the author's statistical analysis, it has been determined that approximately 42.4 percent of door failures have an unidentified underlying cause. Additionally, an estimated 10 percent of these failures can be attributed to issues with the door control unit. These failures may have resulted from undetected software malfunctions. The aforementioned aspects require further investigation, as they demonstrate the potential for utilizing RAMS to evaluate software failures as well. Such motivation is made also in the paper by Kok et al. [2023], which in the pursuit of digital advancement within industrial systems, explores the challenge of applying RAMS (Reliability, Availability, Maintainability, and Safety) methodology, traditionally used in physical assets, to IT/OT converged systems. It provides a case study from the Dutch railway sector to demonstrate the application of a five-step RAMS process on digitized rolling stock design, underscoring the need for improved RAMS analysis tools and multidisciplinary collaboration to enhance the reliability of such advanced systems.

Speaking about the reliability of the software in their comprehensive paper, Immonen and Niemelä [2008] delve into the critical role of software architecture in ensuring the reliability and availability of distributed systems. They argue for the necessity of incorporating reliability and availability analyses from the earliest design stages, introducing a comparative framework that assesses various prediction methods against the backdrop of complex modern systems. Their research identifies key gaps and developmental needs, laying the groundwork for refining these methods to better serve the intricate demands of today's software architectures.

Another study which utilizes the same methodology is Tribble [2002]. This report presents an in-depth examination of the safety analysis of the software used in a Flight Guidance System (FGS), which is of utmost importance in modern aviation. The authors primarily examine the mode logic of the FGS, utilizing formal executable models and a range of analysis techniques, such as Functional Hazard Assessment, Fault Tree Analysis, and Failure Modes, Effects, and Criticality Analysis (FMECA), outlined in Table 4.2, the analysis has focused solely on software failure modes while disregarding hardware failure modes, the exclusion of criticality was due to the similarity in scores. Their methodology relies on Bi-Directional Analysis and model checking, which involves converting requirements and safety properties into formal languages to enable precise analysis. The paper emphasizes the significance of precise and clear software requirements, as inaccuracies in interpretation can result in substantial software problems. This study exemplifies an integrated approach to improving the dependability and security of software in crucial systems, showcasing the efficacy of formal techniques in analyzing software safety.

Failure Mode	Effects	Analysis
Error in FGS-FCP Communications Logic	Incorrect Mode Indication	Flight crew unable to determine mode and state of flight guidance resulting in manual disconnect and manual flying.
Error in FGS-PFD Communications Logic		
Error in "Active / Inactive" Logic		
Error in Mode Selection Logic		
Error in Mode Synchronization Logic		
Error in FCL Selection Logic	Incorrect Guidance	Gradual departure from references until detected by flight crew during check of primary flight data resulting in manual disconnect and manual flying.
Error in "Active / Inactive" Logic		
Error in "Active / Inactive" Logic	Incorrect Indication of Flight Guidance Transfer State	Incorrect "Pilot Flying" side indicated. Possible gradual departure from references until noticed by flight crew during check of primary flight data resulting in manual disconnect and manual flying.
Error in Transfer State Logic		
Error in "Active / Inactive" Logic	Incorrect AP Engagement Indication	If engaged, engagement noticed by resistance to control column / wheel inputs. If disengaged, departure from references noticed during check of primary flight data. Result is manual disconnect and manual flying.
Error in "AP Engage" Logic		

Table 4.2: The Failure Mode Effects and Criticality Analysis for the Mode Logic Level C Hazards Tribble [2002]

Parallel to FTA's and FMEA's roles in enhancing software reliability, Parker and Papadopoulos [2007] presents an innovative approach to optimizing such systems by integrating model-based safety analysis with reliability-cost optimization challenges. This study presents a new approach to multi-objective optimization utilizing genetic algorithms. It highlights the effectiveness of automatically generating safety and reliability models from engineering models and employs techniques such as Hierarchically Performed Hazard Origin and Propagation studies for systematic reliability evaluation. The text demonstrates the effectiveness of model-based safety analysis in identifying the most reliable architectures within predefined cost limitations by comparing constraints-based and Pareto-based optimization.

The paper, authored by La-Ngoc and Kwon [2018], evaluates the efficacy of SFMEA and STPA as safety analysis methods for software-intensive railway level crossing systems. More precisely, the paper conducts a comparison between Software Failure Mode and Effect Analysis (SFMEA) and System Theoretic Process Analysis (STPA). The text explores the significance and complexity of today's software-intensive systems, with a specific focus on ensuring safety. The paper emphasises the challenges faced by safety analysts in selecting

the most suitable analysis technique for particular systems. The paper utilises both SFMEA (Software Failure Modes and Effects Analysis) and STPA (Systems-Theoretic Process Analysis) methodologies to analyse a Level Crossing system. The software-intensive nature of this system and its function in managing railway and road intersections are its defining features. The objective is to avert collisions between trains and other vehicles. The study's findings indicate that STPA is superior in establishing safety requirements at the system level and identifying the underlying causes. On the other hand, SFMEA is valuable for identifying previously unknown hazards at the system level and establishing safety requirements at lower levels. It is recommended to use a combination of both methods to conduct a more thorough safety analysis.

More on safety systems, the study by Miguel et al. [2008] outlines a novel strategy for increasing both the effectiveness and security of software systems in environments where safety is paramount. It suggests a synthesis of safety analysis methodologies with the principles of model-driven architecture to optimize the software development lifecycle, enhancing clearer collaboration between software and safety engineers through a shared modelling language. This approach emphasises the necessity of incorporating safety assessments from the start, refining software architecture, and employing automated tools to maintain consistency and dependability throughout the creation of software systems that are not only cost-efficient but also meet rigorous safety requirements.

Specifically, an assessment of software reliability is done in the paper by Athanasios Kiourtis and Mavrogiorgou [2018], the authors address the challenge of assessing the reliability of Internet of Things (IoT) devices, particularly those with unknown characteristics. They introduce a comprehensive mechanism for evaluating device reliability, encompassing stages such as device recognition, specification classification, reliability estimation, and reliability validation. This mechanism is tested on 21 IoT medical devices, including known and unknown types, using metrics like Inter-Rater Reliability (IRR) and Test-Retest Reliability (TRR). The results indicate the mechanism's effectiveness in reliably classifying and assessing both known and previously unknown devices. Based on the test a calculation for the MTTF, MTTR, MTBF and Availability is done. On these factors, the devices are compared, and evaluations is carried out. This paper contributes significantly to the field by providing a structured approach to evaluating software reliability in IoT devices, a crucial aspect in ensuring the safe and efficient operation of these technologies.

The manufacturing industry's evolution towards customised product offerings necessitates agile and adaptable production systems, sparking interest in Cyber-Physical Production Systems (CPPS) that emphasize modularity and dynamic reconfiguration, this could be said also for the transportation system which seeks to be as effective and as safe as ever. The article by Francalanza et al. [2018] describes a new way to design modular systems for Cyber-Physical Production Systems (CPPS). It uses the Modular Function Deployment method to group products into assembly-oriented families, which makes assembly lines more efficient and adaptable for different types of products. This model also could be potential if adopted by the railway sector. More examples of Cyber-Physical Systems that are used for predictive maintenance is the paper by Lee and Bagheri [2015] delves into the pivotal role of Prognostics and Health Management (PHM) within the integration of computation and physical processes. It underscores the necessity of intelligent systems that can adapt and self-optimize in real time for enhanced productivity and services. The study showcases the successful application of Cyber-Physical Systems (CPS) in predictive maintenance, using industrial robots as a case study, and introduces the "Time Machine Methodology" for organizing big data for PHM, paving the way for a new frontier in maintenance strategies. The CPS model could also be beneficial as a chosen model to assess failure associated with software in the railway sector.

Other aspect of software vulnerabilities that have to be considered are the escalating cybercrime incidents, which saw over 59 million cases in 2015, the paper by de Gusmão et al. [2018] presents an innovative cybersecurity risk analysis model that integrates fault tree analysis (FTA) and fuzzy decision theory. The model checks how weak cybersecurity systems are by looking at possible cyberattack scenarios, what would happen in those scenarios, and the financial and operational effects that would follow. The goal is to provide a structured and quantitative approach to cybersecurity risk assessment, which is usually done with qualitative methods. More on the topic, the paper by Phillips et al. [2018] aims to address the urgent requirement for enhanced practices and methodologies to pursue improved software robustness in space systems, specifically focusing on stressed conditions. This study presents an innovative architectural framework and acquisition approach that aims to enhance the resilience of space system software during the early stages of the system

life cycle. The framework identifies seven essential steps that are crucial for mitigating vulnerabilities and ensuring the success of missions when faced with potential setbacks also the study by Ani et al. [2017] examines the evolution of Industrial Control Systems (ICS) within manufacturing, highlighting their transition from isolated units to advanced, interconnected structures that combine IT capabilities with operational technology. It shows that these systems are more likely to be hacked or messed up by humans. To protect important manufacturing infrastructure, strong cybersecurity measures and training for workers needs to be put in place.

Research has been done directly to address the growing need for reliable software as in Ali et al. [2022] the authors are presenting a novel model that predicts reliability during the design phase of the software, especially for component-based and concurrent software systems. By integrating a new scenario description language and mathematical formulations, the model, which seeks to enhance predictability while managing computational complexity, offers a significant contribution to the field of software reliability analysis. The paper also talks about how they tested their model on two case studies, an ATM system, and an automated railcar system, showing how it can be used for both simple and complex software applications. The results seem promising, as their method could predict software reliability with a lower computational cost compared to other methods, which suggests it could be a useful tool in the real world.

4.4 Partial Conclusion

The literature review reveals various methodologies for evaluating software reliability. Each of the analyzed techniques offers valuable insights for the study, and significant knowledge can be gained when developing the specific approach required by NS. Nevertheless, there is no flawless solution, and it is necessary to employ a minimum of two approaches alongside extensive modification to suit various scenarios. Furthermore, current studies on software reliability assessment only touch upon the general aspects. This suggests that there is plenty of opportunity for further exploration and adaptation to enhance the assessment and comprehension of failures arising from the software component of both IT and OT systems.

Chapter 5

Software Reliability-Centered Maintenance

5.1 Introduction

The literature review in the previous chapter demonstrated that the application of various RAMS methods to evaluate software failures revealed the effectiveness of the Failure Mode and Effective Analysis (FMEA) methodology in modelling software failures. Nevertheless, FMEA is a component of a broader procedure known as Reliability-Centered Maintenance (RCM), as depicted in Figure 5.1. This offers a chance to investigate the feasibility of employing RCM logic to develop a comprehensive procedure for selecting a software system that effectively addresses all potential failures, thereby significantly improving the safety and availability of the system in question.

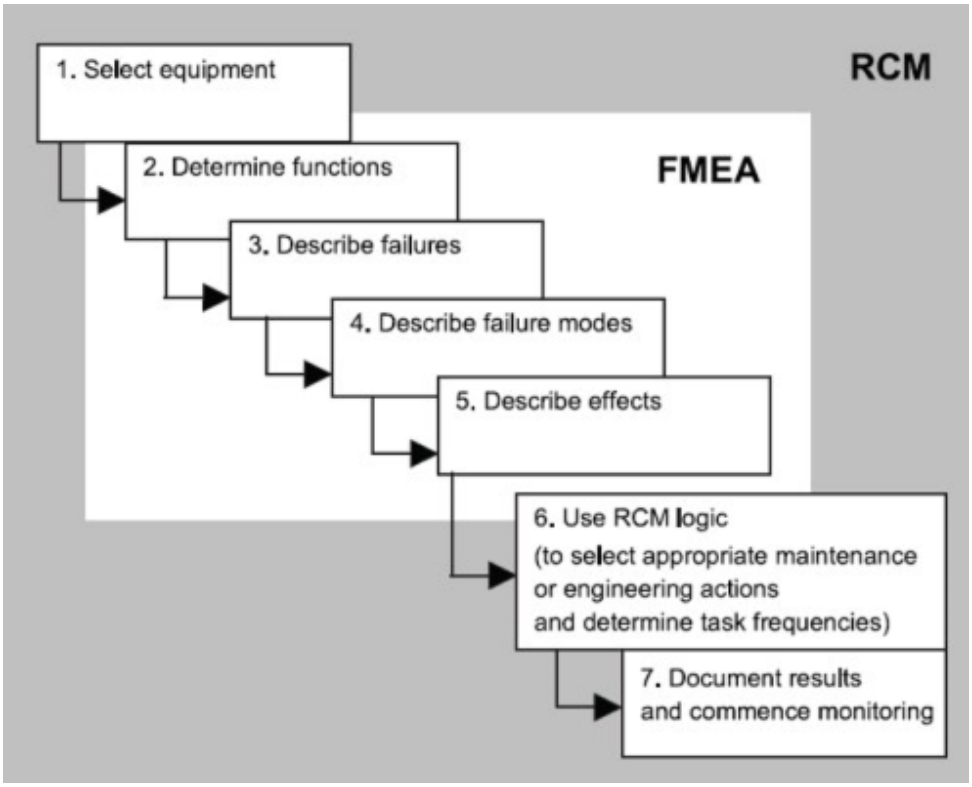

Figure 5.1: FMEA as part of RCM Picknell [1999]

This chapter will outline the necessary modifications required to customise the RCM framework to address software failures. The goal is to bridge the existing gaps and further develop the knowledge in academia and industry regarding methodologies for managing software failures. The main purpose of this chapter is to introduce the RCM logic as a possible solution to model software failure, section **5.2**, present what assistive tools are needed to facilitate this, section **5.3.2**, and what changes in FMEA, section **5.3.3**, and how the maintenance strategies for software failures will differ, section **5.3.4**.

5.2 Reliability-Centered Maintenance

Reliability-Centered Maintenance (RCM) is a methodical strategy for planning maintenance activities that emphasise the reliability of a system or piece of equipment. It is a systematic approach aimed at grasping the inherent reliability of a system, detecting possible issues, and implementing the most efficient maintenance strategies to ensure the system's continuous performance for its intended purpose Siddiqui and Ben-Daya [2009]. RCM encompasses more than just reactive maintenance, it focuses on proactively preventing failures before they occur Moubray [2001].

The RCM process begins with a thorough understanding of the system's functionality and the consequences of failure. This involves a detailed analysis of the system's failure modes, effects, and criticality (FMECA). It follows a certain logic that allows it to be presented in a step-by-step model which aims to provide a guided methodology for modelling potential failures, thus including the FMECA where the goal is to identify how a system can fail, the effects of these failures, and their criticality in terms of safety, operational, and economic impact. By using the RCM logic, the analysis helps in determining the most appropriate maintenance strategies, such as predictive maintenance, preventive maintenance, or run-to-failure Islam et al. [2010], the steps are shown in Figure 5.1. Part of this logic is the utilisation of the RCM logic tree which is a schematic representation that has to guide the decision on what type of maintenance strategy to apply in order to mitigate the failure as effectively as possible. A comprehensive utilisation of RCM logic and an example of such tree can be found in the NASA GUIDE [2008], the tree is shown in Figure 5.2.

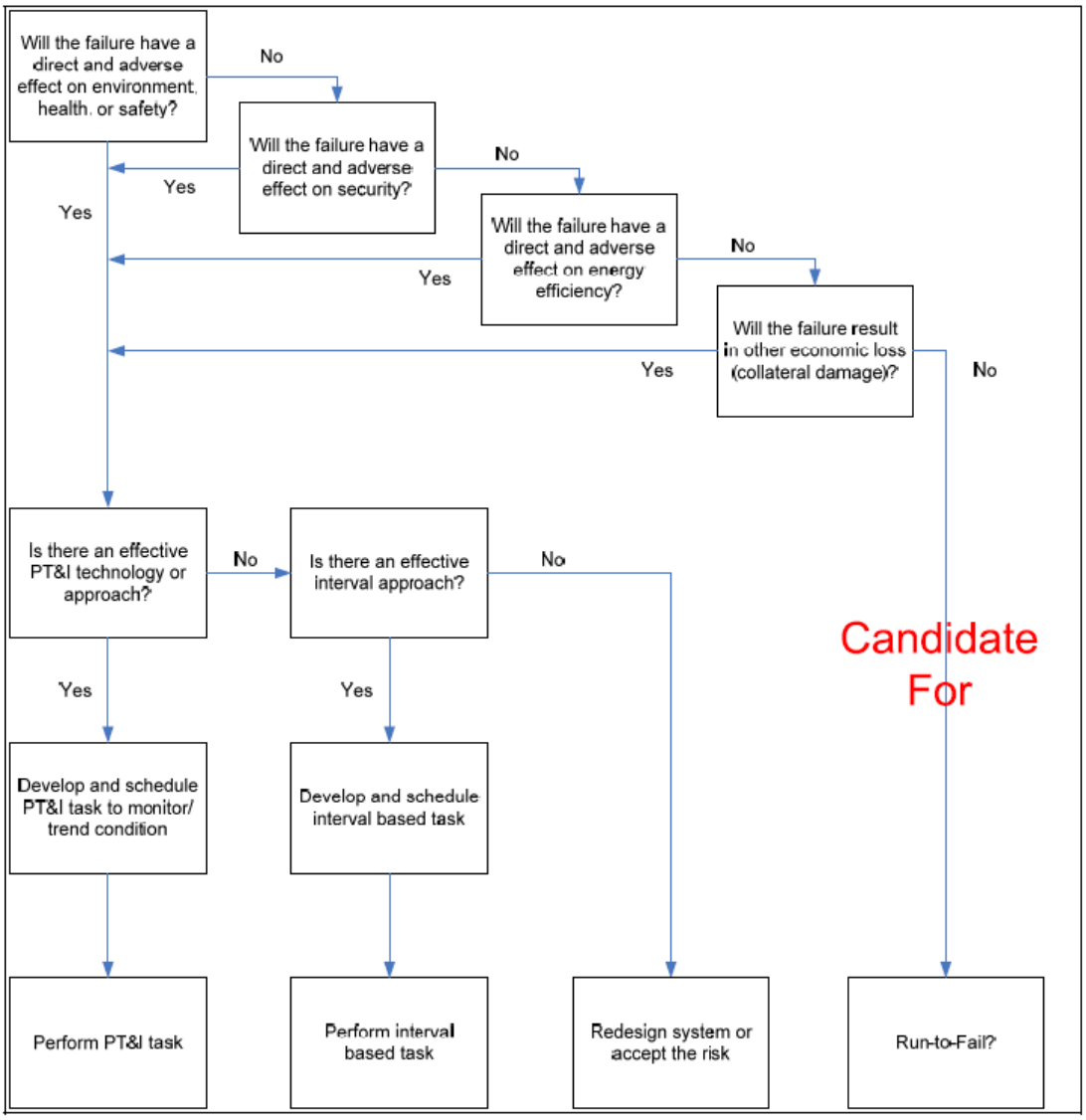

Figure 5.2: NASA RCM Logic Tree NASA GUIDE [2008]

Implementing RCM can lead to significant benefits, including improved system reliability, reduced maintenance costs, and increased operational efficiency. However, it requires a commitment to continuous improvement and a culture that values proactive maintenance. By focusing on maintaining system functionality rather than just fixing broken components, RCM shifts the maintenance paradigm from reactive to proactive, leading to more reliable and efficient operations.

Until now, RCM has primarily been applied to hardware or physical systems. However, there is significant potential for its use in software as well. By modelling software failures, using FMECA, as seen in the previous chapter, it becomes possible to develop a maintenance strategy to effectively address them, when it is also followed by the RCM logic. Nevertheless, the research in that particular field is rather restricted, as the paper by Carretero et al. [2003] represents one of the few attempts to implement RCM on software systems. The primary objective of this research is to address this specific gap in academia. In addition, NS is already familiar with methods such as FMECA, indicating a strong likelihood that they will embrace the RCM logic to enhance the reliability of their software systems. However, some adaptation of this methodology has to be done in order to assess software failures as well. They are all covered in the following sections of this chapter.

5.3 Adaptation of the Methodology

5.3.1 Selecting equipment

The software RCM method begins by first selecting the equipment to be assessed. Here, the choice is made between system-level assessment or component-level assessment. Typically, the selected level is the component level. Hence, the selection of the component must be initiated, beginning with the hardware component, as is typical in RAMS analysis. Subsequently, an evaluation should be conducted to determine if this hardware component interacts with any software. If such an interface is present, an assessment of software failures will be necessary. If the equipment interacts with software, the next step is to select its functions, enabling the initiation of the FMEA process to model potential failures.

5.3.2 Determining the functions of the software

Identifying the functions of hardware components is a relatively simple procedure. However, the situation regarding software functions is not always straightforward, particularly when engineers who have no background in software engineering struggle to comprehend the overall functionality of the software. During the interview phase, several engineers expressed uncertainties regarding the significance of the software functions and their failures, thus providing the foundation of the research about what can be used to ease the understanding of software functions and also be part of the assessment process of its failures. In order to determine the most appropriate tool for this task, extensive research was conducted across various fields.

Firstly, If a look is taken in the software engineering world it will be observed that several graphical models are used when designing a software system. Some of them include:

- **The Unified Modeling Language (UML)** - a general-purpose visual modelling language that provides a standard way to visualize the design of a system. UML offers a standard notation for many types of diagrams, which can be broadly divided into three main groups: behaviour diagrams, interaction diagrams, and structure diagrams Wikipedia-UML.
- **The System Modeling Language (SysML)** - a modelling language specifically designed for the representation of complex systems. It is a derivative of the Unified Modelling Language (UML), which is widely employed in software development. SysML was created to facilitate the modelling of intricate systems methodically, encompassing both the functional and physical aspects of a system WhatisSysML2023. When modelling systems that have hardware and software interfaces the most common way is to use **Block Definition Diagrams**, which are part of the whole SysML family, shown in Figure 5.3.

Secondly, if a look at academia is made in terms of the methods of mitigating software failures, it will be observed that there are modelling methods in use for the software, for example, using different kinds of functional diagrams, data flow diagrams or control flow diagrams. In the paper by Lili [2013], where the authors used these diagrams as assistive techniques to improve the overall process of Software Failure Mode

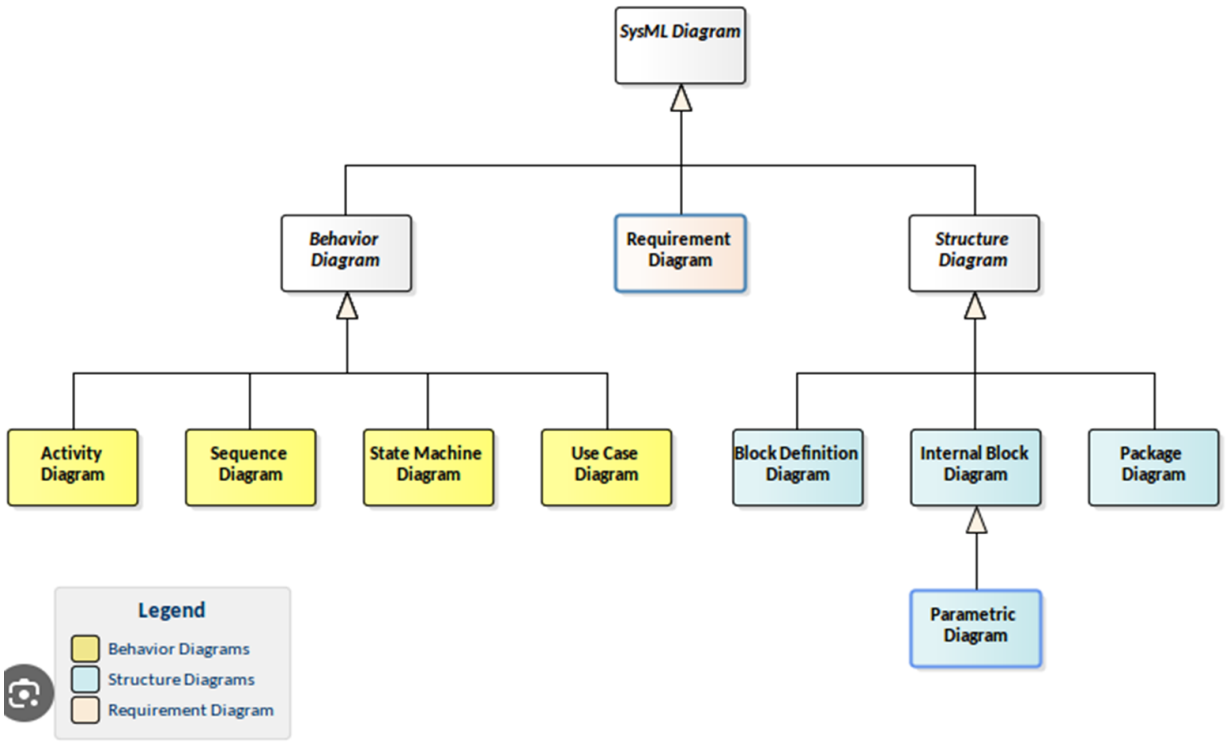

Figure 5.3: System Modeling Language family

and Effect Analysis (SFMEA) and help tracing of failure effects, including local, higher and system-level effects by separating the software in different layers. Another example comes from the paper by La-Ngoc and Kwon [2018], where a Functional Control Structure diagram is used to model the inputs and the outputs of the software and the main functions of a gate in a Railway Level Crossing System, seen in Figure 5.4. Such diagrams are quite simple to review, thus enhancing the understanding of how software systems are functioning for the people who are further away from the software engineering field.

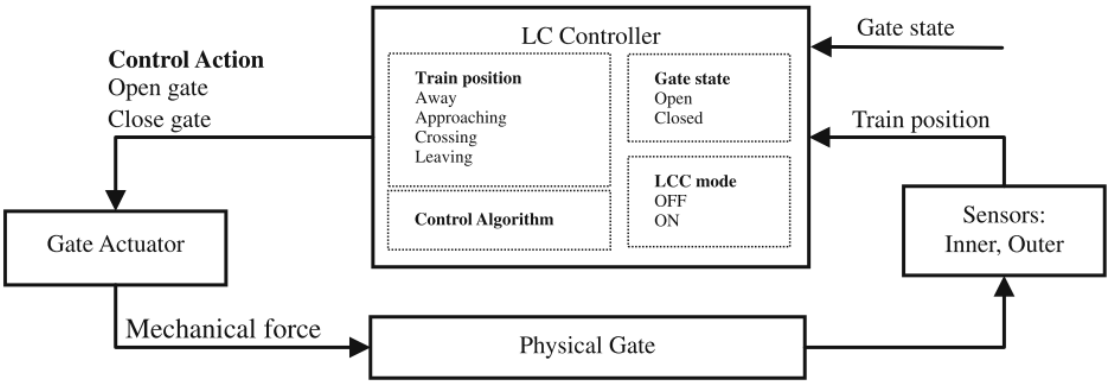

Figure 5.4: LC Functional Control Structure diagram. La-Ngoc and Kwon [2018]

Finally, regarding the latest standard in the field of Railway Software Development, CENELEC EN 50716:2023 universally recommend creating a comprehensive model of the software system as an important step in the design process. This practice is particularly essential for ensuring clear communication and understanding between the software supplier and the organization commissioning the system, as is the case between NS and its suppliers. The standard clearly states that by providing a detailed and visual representation, the model acts as a common language that bridges any gaps in terminology or perspective, thereby reducing the risk of misinterpretation or ambiguity, as observed in Table 5.1.

TECHNIQUE/MEASURE	Ref	Basic Integrity	SIL 1	SIL 2	SIL 3	SIL 4
1. Data Modelling	D.65	R	HR	HR	HR	HR
2. Data Flow Diagrams	D.11	-	HR	HR	HR	HR
3. Control Flow Diagrams	D.66	R	HR	HR	HR	HR
4. Finite State Machines or State Transition Diagrams	D.27	-	HR	HR	HR	HR
5. Petri Nets	D.55	-	HR	HR	HR	HR
6. Decision/Truth Tables	D.13	R	HR	HR	HR	HR
7. Formal Methods	D.28	-	HR	HR	HR	HR
8. Performance Modelling	D.39	-	HR	HR	HR	HR
9. Prototyping/Animation	D.43	-	R	R	R	R
10. Structure Diagrams	D.51	-	HR	HR	HR	HR
11. Sequence Diagrams	D.67	R	HR	HR	HR	HR
12. Cause Consequence Diagrams	D.6	R	R	R	R	R
13. Event Tree Diagrams	D.22	-	R	R	R	R
Requirements: 1) For SIL 1-4, modelling guidance shall be defined and used. 2) For SIL 1-4, at least one of the HR techniques shall be chosen.						

Table 5.1: Modelling of software systems EN 50716:2023

This alignment is vital for accurately capturing the organization's (NS) requirements and translating them into a functional and effective software solution. Additionally, a well-defined model facilitates a better understanding of the already designed product for the organization, which enhances transparency and allows for more precise planning and resource allocation. Ultimately, adhering to these recommendations and developing a comprehensive model can serve as a fundamental measure in the effort to reduce software failures. This is because gaining a thorough understanding of a system's functionality allows for a more accurate prediction of its potential failures.

These diagrams are well-suited for integration into the software RCM process. They can be utilised for representing the outputs, inputs, and functions of the software. Next, the various software layers will be derived from the functions. The layers and the functions will serve as an input for the SFMEA, where the potential failures for each layer will be modelled. This step allows for the entire analysis to be scaled. The model offers the choice to select general functions for the assessment process, which keeps it at a general level. However, if the system is safety critical or experiencing a high number of failures, more specific functions can be chosen. This will result in more comprehensive modelling of failures in the SFMEA, thereby improving the overall modelling process. To showcase a possible diagram, a simple example of what such a diagram could look like for the ERTMS system is derived in Figure 5.6.

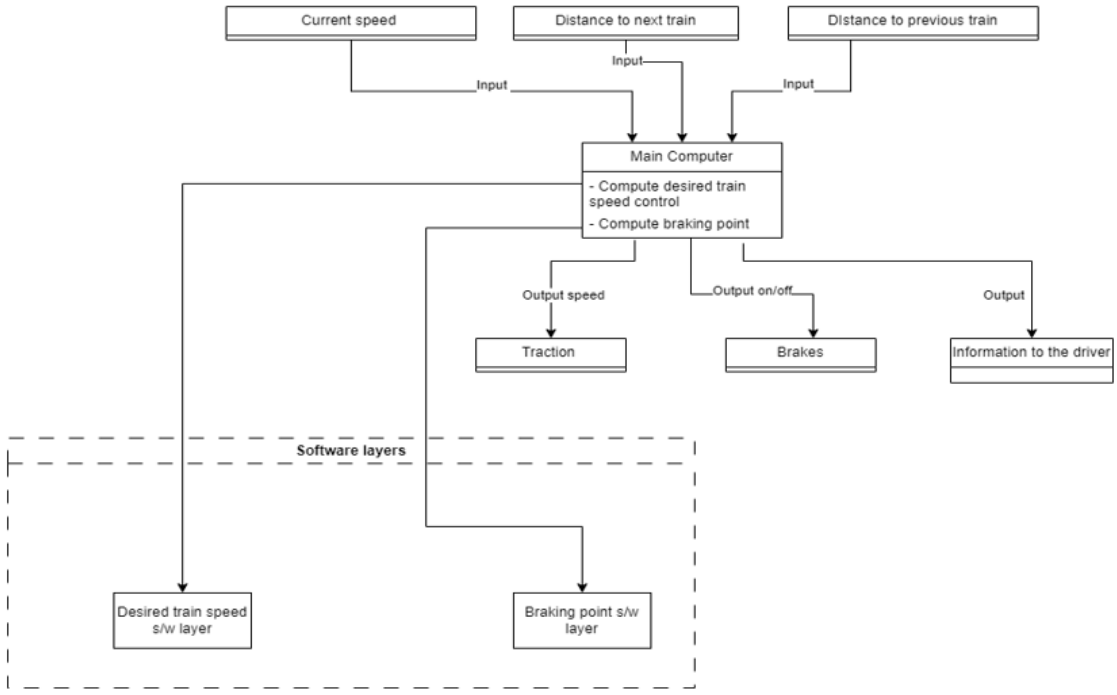

Figure 5.6: Example of a simplified block definition diagram for ERTMS

5.3.3 Software FMEA

Once all the functions of the desired software system have been identified, the next stage of the Software RCM involves conducting a Software Failure Mode and Effect Analysis (SFMEA). The potential failure modes are modelled along with their corresponding effects and causes. This step follows the existing FMEA process that is already being used by NS and its suppliers, but of course, here software failures are modelled rather than hardware ones. The primary distinction between regular FMEA and SFMEA is introduced in the selection of the failure's cause. To streamline the model and account for the wide range of potential causes for software failures, a classification is provided that aligns with the one outlined in the standard IEC 62628:2012. The software faults are classified into five distinct categories:

- Specification faults
- Design faults
- Programming faults
- Compiler-inserted faults
- Faults introduced during the software maintenance

This classification assists the user in selecting the appropriate root cause of the failure. Nevertheless, there is a lack of high efficiency in this task, because it does not explicitly indicate the specific reason. However, categorising each failure into separate groups can assist in identifying the actual cause of subsequent software malfunctions.

Then the SFMEA again follows the step of the regular one, meaning that all of the failures have to be scored in their Severity, Occurrence and Detection with scores from 1 to 10. Then the Risk Priority Number (RPN) is calculated using the following formula:

$$\text{RPN} = S \times O \times D$$

where:

$$S : \text{Severity}$$
$$O : \text{Occurrence}$$
$$D : \text{Detection}$$

The RPN helps prioritize the failures and steer the decision within the logic of RCM for choosing the right strategy for mitigating every software failure.

It is worth noting that, as mentioned in the literature cited above, Software Fault Tree Analysis (SFTA) is commonly used alongside SFMEA in most of the papers. SFTA is a top-down method that enables the identification of the underlying cause of each failure. The method is highly effective because it enables the easy modelling of all potential root causes of the failure through its graphical representation of the tree. However, because of time limitations, this research will only recommend future research for integrating SFTA into the SRCM logic, as the methodology is too complex to be thoroughly researched and implemented.

5.3.4 Depicting maintenance strategy to mitigate the failure

Upon completing the SFMEA tables and accurately assessing and ranking all potential failures in the provided software system, the next step is to proceed to the SRCM logic subsequent stage, which involves selecting the most suitable maintenance strategy to minimise the risk of failure.

5.3.4.1 Software Maintenance strategies

The most commonly employed maintenance strategy for software failures is the corrective approach Carretero et al. [2003]. When a software failure, such as a programming code bug or incorrect configuration setup, is identified, an update is released to correct the failure. This strategy is also applied to hardware, whereby corrective action is taken when a component malfunctions, such as repairing or replacing the faulty hardware component, in order to address the problem.

Another frequently employed approach is the strategy of preventive maintenance. This implies that the anticipated failure is acknowledged and the software is modified to mitigate the impact of such failure, resulting in a less severe consequence compared to previous occurrences. This approach can also be applied to software failures, by modifying the software to mitigate the impact of such failures. An illustrative instance of adaptive maintenance for software is the situation that occurred with the Airbus A330 following the Qantas Flight 72 incident ATSB [WA, 7 October 2008, VH-QPA Airbus A330-303]. Following an investigation into unknown software issues that resulted in a hazardous nose-down movement of the aircraft, the root cause of the problem remained undetermined. However, the software was subsequently changed to prevent the aircraft's main computer from autonomously performing the same dangerous manoeuvre in the event of a recurrence. In terms of hardware, proactive replacement of a line replaceable unit is typically carried out to avoid its failure during operation Basri et al. [2017]. Hardware component wear or complete failure can lead to software faults. Taking preventive measures for the hardware component will also prevent software faults. An instance of this can be illustrated by a situation where a sensor experiences a minor malfunction, causing it to still operate but provide different inputs to the software. Consequently, the software, not being programmed to handle such inputs, fails to execute its logic successfully. Initially, this issue is attributed to a software failure, but upon further investigation, it is determined that the underlying cause is a hardware component. This indicates that implementing preventive maintenance for the hardware can also help to reduce software failures.

However, as stated by Swanson [1976], software maintenance encompasses two additional dimensions, adaptive and perfective maintenance. Adaptive maintenance is a strategy that focuses on modifying software to accommodate changes in the application system. This approach is implemented when there is a modification in either the application or the hardware that directly interacts with the software. In such a scenario, the software needs to be adjusted to align with the altered environment, in order to prevent future unwanted failures. This is typically true in IT systems, where technology advances rapidly and various hardware components are upgraded over the system's lifespan. Nevertheless, the software must continuously accommodate each new hardware modification, necessitating adaptation to the dynamic environment.

Conversely, perfective maintenance focuses on improving performance, implementing new programme features, and enhancing the program's future maintainability. It is utilised when the software system's performance is unsatisfactory and requires modifications to reach its maximum capability. The main advantages include improved efficiency, enhanced user satisfaction, prolonged software longevity, and diminished expenses for future maintenance.

The final software maintenance strategy is the "Run-to-failure" approach. This strategy is widely recognised in the field of hardware maintenance. It involves operating a component until it malfunctions, based on the calculation that this is the most cost-effective approach. Nevertheless, within the domain of software maintenance, the concept of "Run-to-failure" carries a distinct interpretation. When a software system is allowed to fail, it typically means that this failure does not significantly affect the overall performance or safety of the system. This is due to the occurrence of numerous phantom software failures, which can be triggered by an unforeseen sequence of inputs for the software. These failures are falsely flagged as errors, when in fact no actual faults are present. If a failure is accurately modelled and its risk is assessed to have no impact on the system when it happens, the Run-to-failure strategy can be employed. This implies that the maintenance crew and anyone interacting with the system are disregarding this failure.

To summarize, five existing maintenance strategies can be applied to the software systems:

1. Corrective - a detected software fault is mitigated by implying a software, firmware or configuration update;
2. Preventive - an update for the software is done to avoid a detected potential failure or a hardware component that has a direct input for the software is preventively replaced to avoid wrong input;
3. Adaptive - an adaptation of the software is done to suit the new environment that it is operating in;
4. Perfective - an update of the software is introduced to improve the system performance;
5. Run-to-failure - the software failure is completely ignored and the messages that are flagged are deleted.

5.3.4.2 Software RCM logic tree

The Software Reliability-Centered method employs the RCM logic tree to ask the appropriate questions and accurately depict the desired strategy. This is an illustration of the sequential steps that need to be followed in order to select the strategies mentioned earlier.

The tree must exhibit a seamless and non-intimidating user experience. However, it must be sufficiently effective in its role of providing guidance for the most optimal approach to address the currently evaluated failure.

After conducting a comprehensive overview of the RCM logic tree used in various industries, including the NASA RCM logic tree illustrated in Figure 5.2, the SRCM logic tree depicted in Figure 5.9 was created and is suggested to be placed in use. It is appropriate for the specific requirements of NS and the operational systems used in the railway industry. Nevertheless, this tree is susceptible to modifications with respect to the software system that will be evaluated in order to generate superior outcomes.

Upon examining the tree, it is evident that the primary concern revolves around the safety of the software system. The user should prioritise determining whether the software failure is safety-critical or poses a cyber-security risk to the train's operation. If the failure is not deemed to have a significant impact on safety, then the train's availability becomes a viable consideration, making it a secondary option to be considered. If this is not true, then it should be verified whether perfective maintenance can be applied to the software to enhance the system's performance. If this is not possible, then the failure can be disregarded.

If it is determined that this failure has a significant impact on safety or greatly affects the train's availability, a check should be conducted to determine if a simple update can resolve the issue. If this is not feasible, then an assessment should be conducted to determine whether a modification in the software's operation or the requirements set by the organisation and the supplier is necessary, in this case between NS and its suppliers. Occasionally, a minor alteration in the functioning of the entire system or a single modification in a document containing a set of requirements may successfully mitigate a software failure. However, if it is not feasible or economically viable, a proactive approach must be implemented either on the software or hardware side to minimise the potential impact of the software failure. The preventive approach means that the potential failure and its causes are fully acknowledged and the right modification of the software is pushed to make it impossible for such failure to appear in the operational phase.

5.3.5 Overview of the Software Reliability-Centered Maintenance

An overview of the adapted framework of the RCM logic is shown in Figure 5.7. The author of this thesis created this overview as an inspiration was taken by the one shown in Figure 5.1, which is used for hardware components. The purpose for this was that both can be directly compared and the differences between regular RCM and software RCM can be directly drawn. Upon initial examination, a clear parallel can be drawn between the two. However, it becomes evident that the utilisation of assistive methodologies, such as functional diagrams, is necessary in order to accurately outline the functions and layers of the evaluated software system, as the SFMEA utilises them as inputs. Using the SFMEA methodology mirrors the standard process, however, some modifications should be made to the process of modelling potential failures, such as categorising software faults, to facilitate the selection of the fault's cause.

In addition to SFMEA, another methodology called SFTA can be employed to identify the underlying cause of the failure. However, it should be noted that this is merely a suggestion and not an integral part of the initial implementation of RCM.

The user must ultimately choose the best strategy, which an SRCM logic tree will help with. This logic tree will assist the user in choosing the most suitable maintenance strategy to successfully mitigate the assessed software failure. A complete overview of the flow of modelling software failures is made in Figure 5.8. The suggested method tries to achieve the second research objective of how software failure can be incorporated into the RAMS process in order to facilitate their successful mitigation.

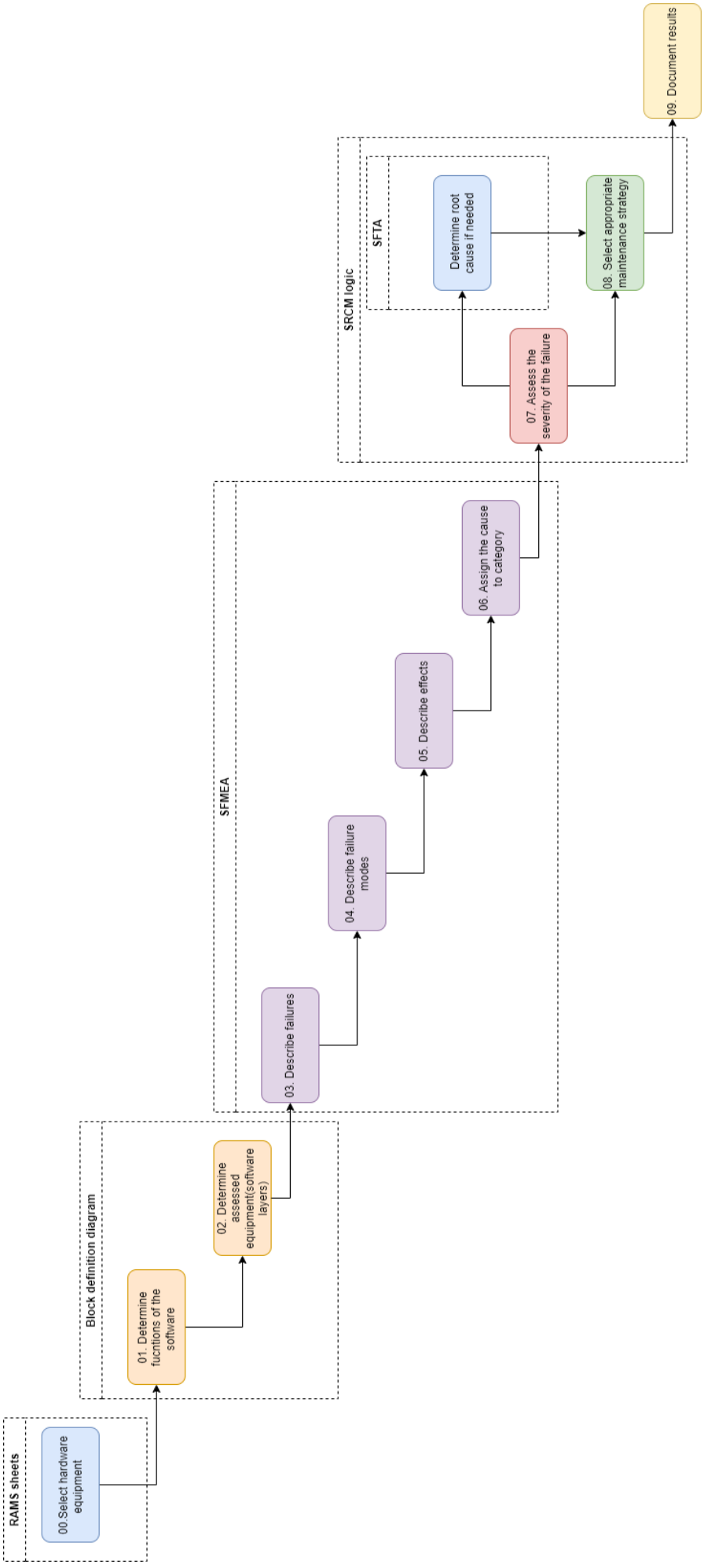

Figure 5.7: Software Reliability-centered maintenance adapted steps

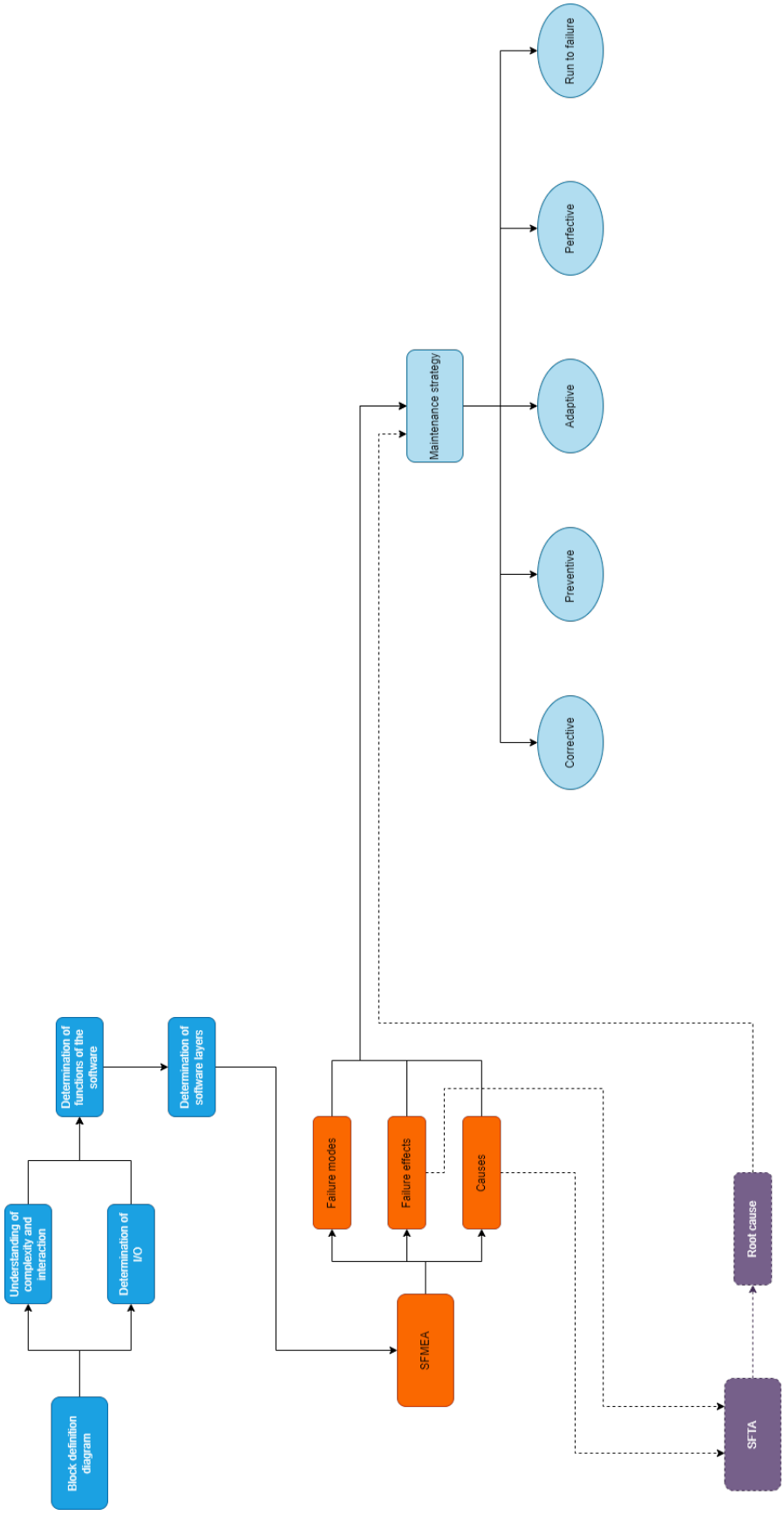

Figure 5.8: Software Reliability-Centered maintenance model flow

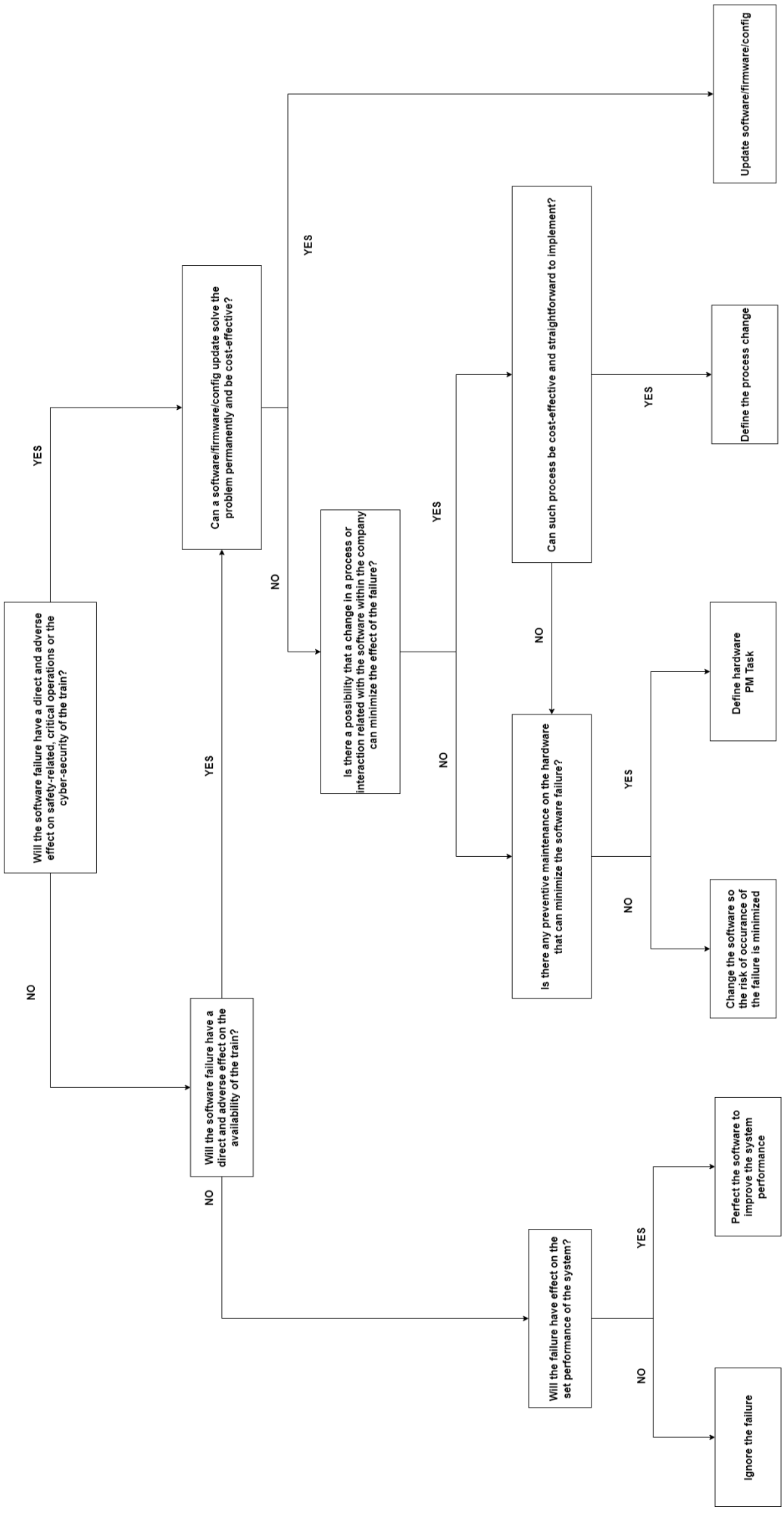

Figure 5.9: Software Reliability-Centered maintenance logic tree

Chapter 6

Modelling of software failures

6.1 Introduction

To utilise and showcase how all of the proposed ideas in the previous chapter can be implemented in the realm of maintenance, an Excel tool was developed based on already existing ones within the Dutch Railways (NS). This chapter will introduce this tool, which is designed to model and analyse potential software failures. Also, the structure and functionality of the Excel model, along with its underlying assumptions, will be presented. Moreover, this chapter aims to demonstrate the model's utility as a possible vital component in the proactive management of software failures in the railway industry.

6.2 Excel model

6.2.1 Main idea

The developed Excel tool is based on the FMEA template that is established and used within NS and its suppliers. The main reason behind it is to be as familiar as possible for the engineers that are already accustomed to such a process, so it does not cause unnecessary confusion when it is introduced. That is also important, as in the case of NS, the FMEA is usually filled by the supplier of the system/component. That means that the chance of all of the various suppliers of NS using this new tool for assessing software failures is higher when it is similar to the templates they already know well.

The tool is created to follow the steps of SRCM logic and be as intuitive for engineers who have background knowledge of the processes of RCM and FMEA. However, it has to be mentioned that the knowledge of the additional theory presented in the previous chapter of what adaptations are present in the logic of RCM to assess software failure needs to be obtained to ensure that every step of the Excel tool is rightly understood and followed for the maximum efficiency of the final goal - the mitigation of potential software failures.

6.2.2 Choosing the component

So the process starts with selecting the software component that has to be assessed within the system. In the case of NS, the RAMS sheets that are already in use can be utilised for this purpose as well, their hardware components of a system are already depicted, so just another two columns have to be added for the user of the RAMS sheets to ask if this hardware component has any interface with software. if this is the case, then software failures will be present and they have to be assessed. Therefore, the cell will be illuminated in red, and a link will be created to the Excel file where the tool for assessing software failures is located, so when it is clicked, the user will be guided to the tool. This is shown in Figure 6.1.

LRU Description	Part Number	Qty	Function	Software interface	Software failure assessment
VNAS4200-X		1,00	Video recorder	YES	Needed - Click to assess
VNASSTOR6000-SA1		1,00	Media storage	NO	Not needed

Figure 6.1: Example of selecting software interface via NS RAMS sheets for the CCTV system.

6.2.3 Modelling software functions and layers

Upon clicking on the red cell in the RAMS sheet, the user will be transferred to the first sheet of the developed Excel model. The purpose of this sheet is to introduce the user to the process of using the tool and model all the software functions and failures, as well as how to pick the right strategy to mitigate them. The model is divided into four main steps:

1. Filling the SFMEA cover sheet, which includes filling a Functional block diagram, where the software input/outputs and functions are modelled.
2. Filling SFMEA sheet.
3. Scoring every failure - RPN
4. Choosing the appropriate maintenance strategy using the RCM logic tree

The layout of the main steps of the tool from the guidance sheet is shown in Figure 6.2.

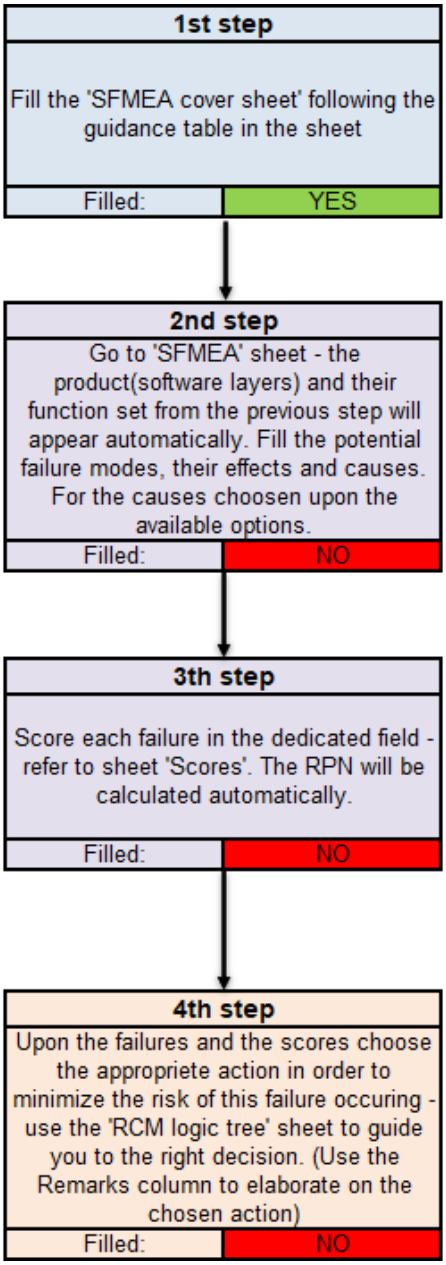

Figure 6.2: Model guidance sheet.

Proceeding into the first step of the tool - filling the SFMEA cover sheet. This is a well-known process for carrying out an FMEA analysis. In the hardware failure assessment, this sheet includes several tables that serve to get the information for the team that is filling out the FMEA sheet itself. For software failures, some changes are introduced, as this sheet is a good place to do the modelling of the software inputs, outputs and functions. Also, the chosen functions have to be divided into different layers or sections. Figure 6.3 depicts an "Example Functional Block Diagram" that aids the user in accomplishing that. Next to the diagram is a table, where automatically the LRU description and its functions will appear, which were clicked previously in the RAMS sheets. Below them, the user will have to fill in the inputs, outputs, functions, and layers of the software, Figure 6.4. This will help with the initialization of the functions of the software. In this step, the model can be scaled by choosing the level of functions of the software – it could be more generalised or in greater detail, which will greatly influence the outcome of the model and can steer the whole assessment.

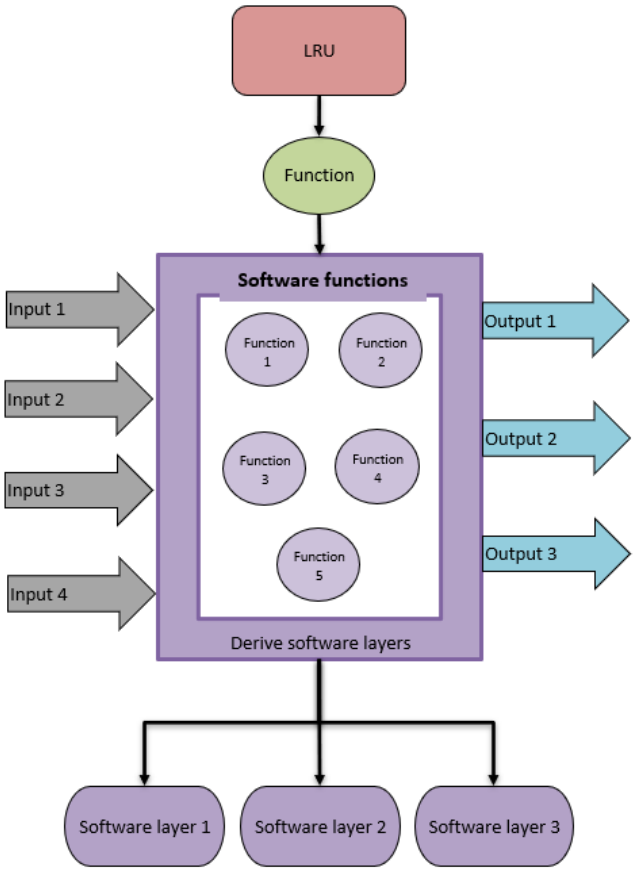

Figure 6.3: Example Functional Block diagram for the guidance of the user.

Type FMEA:	Software FMEA
Software Functional Analysis	
LRU	
0	
Function	
0	
Define Inputs	Define Outputs
Define Software functions	Define Software Layers

Figure 6.4: Table for filling software inputs, outputs, functions and layers.

6.2.4 Modelling software failures

The second step of the Excel tool takes the user to the SFMEA sheet, where he/she is prompted to carry out the Software Failure Mode and Effect Analysis, Figure 6.5. This sheet is pretty much unchanged from the original FMEA template that NS uses for modelling hardware failures, so the user should be familiar with how to navigate within it. However, the main difference is that the 'Product Description' will be the software layer chosen in the previous step and its function accordingly. This will appear automatically. Therefore, the user only needs to fill in all the potential failure modes and their effects for every software layer chosen previously. Then the other difference from the original template comes in the choice of the possible cause of the software failure. As stated in the previous chapter, the software faults are categorised according to standard CENELEC IEC 62628:2012. A drop-down menu will appear from which the choice will be made, Figure 6.6. This will ease the user when choosing the cause of the failure and will put the failure into the set category, which will help to mitigate it later.

Product description level 1	Function	Potential Failure Mode	Possible effects of the failure	Severity	Classification	Possible causes of the failure

Figure 6.5: SFMEA table

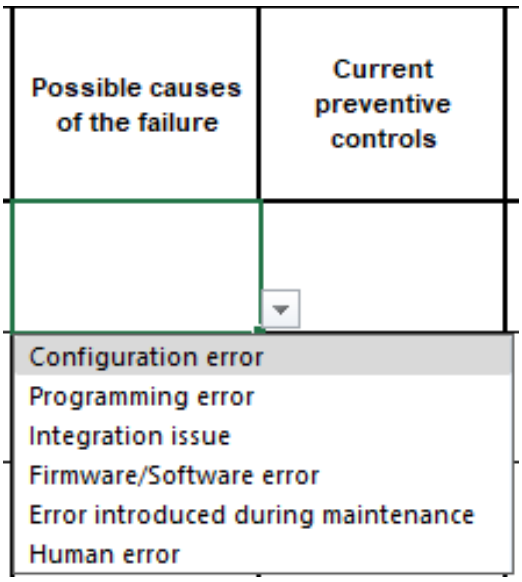

Figure 6.6: Possible cause of failure drop-down menu.

After all the possible failures and effects are modelled into the SFMEA table, the user has to put scores from 1 to 10 for their severity, occurrence, and detection. For this, the Risk-priority number for each failure is automatically calculated. This process follows the original one from the hardware FMEA.

6.2.5 Choosing maintenance strategy

After all of the above steps are finished, this means that the SFMEA is done, and following the SRCM steps, Figure 5.7, the next step is to select the most appropriate maintenance strategy to mitigate the software failure. To guide the user, the SRCM logic tree, presented in the previous chapter, Figure 5.9, is placed in the last sheet of the Excel tool. The user has to go through it and select the most appropriate maintenance strategy. This is done at the end of the SFMEA sheet with a drop-down menu, Figure 6.7, where all of the strategies resulting from the tree are available for selection. The column where the user has to pick the strategy is named 'Action', which follows the original template, also next to it is the 'Remarks' column, where the user can specify the exact maintenance action or add some comments for the already chosen one.

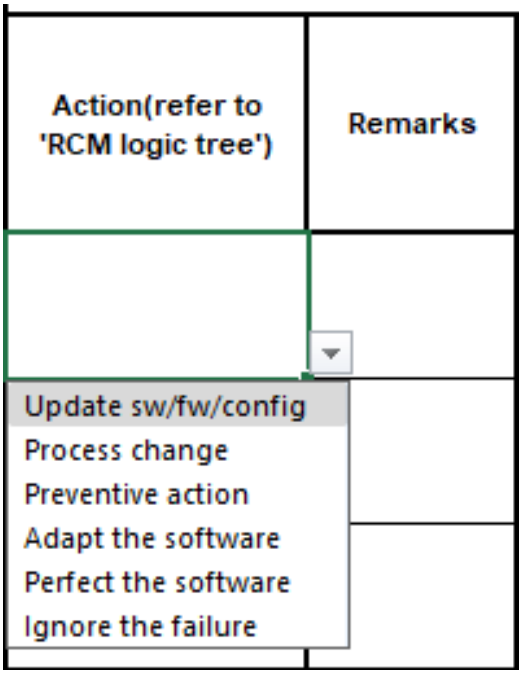

Figure 6.7: Maintenance strategies drop-down menu choice.

With the choice of the maintenance strategy and adding the additional remarks if needed, the user is set to finish the final step of the model. This includes scoring the failures once again, but now with the adjusted scores that are expected after the strategy is applied. Thus, a direct comparison can be made between which improvements will be made and how effectively the failures are expected to be mitigated.

6.3 Use case

To enhance comprehension of the Excel tool that was created, a use case is illustrated to demonstrate the application of the methodology. The train's Closed-Circuit Television (CCTV) system was selected due to its inherent susceptibility to software malfunctions during operation Kok et al. [2023]. In order to streamline the case, only one component was selected from this system, specifically the Network Video Recorder (NVR) depicted in Figure 6.8. A Network Video Recorder (NVR) is a central component in an IP-based video surveillance system, designed to record and manage video footage captured by network cameras (IP cameras). Unlike a Digital Video Recorder (DVR), which works with analogue cameras, an NVR is built to handle digital video streams directly from IP cameras over a network.

Figure 6.8: Overview of a NVR

The analysis commences by illustrating the inputs and outputs for the software of the NVR. Subsequently, the primary functions are derived and integrated within various software layers, Figure 6.9. It should be noted that illustrating these functions and layers can influence the analysis in various ways. Greater availability of knowledge, expertise, and data regarding the functions and failures of a given component allows for a more precise depiction of its functions, leading to a more thorough analysis of failures. This, in turn, improves the overall output of the model. Nevertheless, even with limited knowledge of the analysed component, it is still possible to create a model of the basic function and potential failures. This will allow for a more general analysis, which in turn will impact the maintenance strategy for the entire system.

Type FMEA:	Software FMEA
Software Functional Analysis	
LRU	
NVR	
Function	
Video recording	
Define Inputs	Define Outputs
Video from Camera	Video to Recording device
Temperature	Video to Streaming device
User interface input	Cooling control
	User interface output
Define Software functions	Define Software Layers
Recording video	Recording and Storage layer
Storage management	
Playback video	
Live streaming	Streaming and communication layer
Network and communication management	
Diagnostics and Monitoring	Self-diagnostic and UI layer
User Management	

Figure 6.9: Functional analysis of the NVR

Product description level 1	Function	Potential Failure Mode	Possible effects of the failure	Severity	Classification	Possible causes of the failure	Current preventive controls	Occurence	Current detection controls	Detection	RPN	Action(refer to 'RCM logic tree')
Recording and Storage layer	Recording video	Camera rotation settings not applied correctly	Recording is at unsuitable angle	4		Integration issue		6		2	48	Process change
	Storage management	Crashes at high bit rates with big video frames	Corrupted video data	6		Programming error		5		3	90	Perfect the s/w
	Playback video	Recorder does not operate at sub-zero temperatures	Not possible to watch recorded videos	6		Configuration error		5		3	90	Update fw/config
Self-diagnostic and UI layer	Diagnostics and Monitoring	High CPU load	Spurious resets of the device	8		Firmware error		3		6	144	Update fw/config
	User Management	Wrong timestamp and session ID handling in metadata	Not possible to review the exact recording	2		Programming error		4		1	8	Ignore the failure
Streaming and communication layer	Live streaming	Wayside live stream using the wrong channel in IPT	Not possible to watch live videos	4		Configuration error		6		2	48	Perfect the s/w
	Network and communication management	Issue with network intefaces not being correctly configured	No connection present	7		Configuration error		7		3	147	Update fw/config

Figure 6.10: Software FMEA of the NVR

The functional analysis provides the input for the Software Failure and Effect Analysis (SFMEA) by identifying the software layers and their corresponding functions. The potential failure modes and their effects for this use case were determined based on the information provided in a document containing the update notes from the supplier of the CCTV system for the Dutch Railways. Thus, these failures have already been encountered and resolved for the NVR. However, they continue to offer a valuable chance to showcase a potential future utilisation of the model.

Once the potential failures were identified, the possible causes were selected from a drop-down menu and each failure was assigned a score. For this particular case, the purpose of these scores is not to accurately reflect the severity of each failure, but rather to indicate how they can impact maintenance actions. For failures with the highest Risk Priority Number (RPN), it is generally advisable to promptly address the issue

by implementing necessary updates. Furthermore, by examining the very first failure, it is possible to identify instances where the camera setting is not configured correctly. In order to proactively reduce the likelihood of such failures, it may be beneficial to modify the procedure for selecting and implementing these settings. Nevertheless, it must be emphasised that the illustrated potential failures and their effects for this particular scenario may not accurately reflect the actual situation.

It is important to note that in certain situations, even if the Risk Priority Number (RPN) is high, as it is for the second and third failures, if they can be easily detected and resolved with a quick and inexpensive update, no preventive action is necessary. However, it is important to be aware that these failures can still occur during the operation of the software. Another distinction between hardware FMEA and software FMEA is that correcting hardware failures is often more costly and time-consuming than correcting software ones.

Another instance of failure can be observed when an incorrect timestamp is applied to the video, second to last failure. This occurrence may not affect the performance of the NVR and could happen sporadically, making it unworthy of attention and it is possible to be disregarded. Occasionally, insignificant software updates can unintentionally disrupt crucial functions, so the chance of this happening also should be limited by limiting unnecessary updates.

Additionally, this use case demonstrates that when designing a new system, the data obtained from past software failures in similar systems can be valuable when completing the SFMEA (Software Failure Modes and Effects Analysis) for the new product. Typically, there is a greater amount of recorded data for software failures compared to hardware failures. The management of this data will be challenging and could require innovative solutions, such as the use of Artificial Intelligence, to effectively sort and assist in filling the SFMEA.

6.4 Partial Conclusion

The Excel tool was introduced and offered to NS as a suitable foundation for introducing Software Reliability-Centered maintenance in the company's failure assessment process. The tool's design is deliberately crafted to closely mimic the existing FMEA tools, ensuring that it is not too daunting or challenging for the engineers who are expected to use it, including all of NS's suppliers of systems and technologies. The goal was to develop an evaluation procedure that closely resembles the ones mandated by NS for its suppliers, with the aim of increasing the probability that all of NS's suppliers will adopt it. This will facilitate the identification of potential software errors at the design stage and enable software system suppliers to suggest maintenance techniques to NS engineers for mitigating these malfunctions throughout the operational phase of the software systems.

Chapter 7

Evaluation and Results

7.1 Introduction

An explanatory video was prepared to assess the developed Excel model. The submission comprised a screen recording that commenced with a concise explanation of the concepts of SRCM and followed the logical progression of the assessment on paper, as detailed in Chapter 5. Next, the viewers became familiar with the Excel tool, and all the characteristics and procedures of the model were demonstrated, mirroring Chapter 6. Subsequently, a survey was constructed of fourteen questions, designed to assess the model.

7.2 Results

All the engineers who were previously interviewed in this research were provided with the video and survey to complete since they were already informed about the ongoing research inside NS. A total of fifteen individuals were requested to see the film and provide their separate assessments via the survey. As a consequence, eleven individuals participated in the survey. Although this sample size is not regarded as statistically significant in terms of scientific value, it is worth noting that the absence of statistical significance does not necessarily mean that the results should not be considered relevant Wasserstein and Lazar [2016]. In addition, the survey respondents consisted of engineers from many domains, as shown in Table 7.1, so enriching the range of viewpoints and perspectives obtained. The tables below present all the outcomes obtained from the questionnaire. The final section of the survey was an open-ended question where participants may provide 'Additional Remarks'. The survey findings from the open question are provided in the Appendices.

Position	Count
System Engineer	3
IT Engineer	1
Senior Engineer	1
Lead Engineer	1
Cluster Lead	1
Aspect Manager	3
Maintenance Engineer	1

Table 7.1: Positions of the interviewees within NS

The data was gathered anonymously, and the engineers were provided with 4 or 5 distinct choices that corresponded to their degree of agreement with the posed issue. The initial question was to gather their perspectives on the concept of employing adjusted RCM logic for modelling software failures, as indicated in Table 7.2. The results indicated that, while opinions were mixed, they rated the notion as sufficiently potent and appropriate for the desired objective.

How do you find the idea of applying adapted RCM logic to model software failures?	**Count**
Not suitable at all	0
Some of the ideas are suitable	4
Suitable but more changes are needed	2
Really suitable and has good potential	5

Table 7.2: Opinions about the whole idea

The subsequent two questions related to the user experience, as indicated in Table 7.3. It is important to mention that none of the engineers personally tested the model at the time of writing of this report. However, a video demonstration that faithfully replicates all the actions a user would take informs their opinions. Once again, there was a significant divergence of viewpoints, with strong polarisation about the ease of use. This might be attributed to the diverse range of engineers that participated in the poll. Individuals who had a higher level of knowledge regarding the RAMS process, particularly concerning FMEA, reported finding it simpler to adhere to this model.

What do you think about the model structure?	**Count**	**Do you find the model easy to use?**	**Count**
It definitely needs several changes	2	It is hard to follow with a steep learning curve	0
It needs minor changes	0	It is hard to follow but still manageable	1
Acceptable, but could be improved	2	It is moderate in terms of flow and usage	3
Good, but there is still room for improvement	7	It is easy to follow but hard to learn to use	4
The model layout is perfect	0	It is easy to follow and learn to use	3

Table 7.3: User Experience

The subsequent five questions asked the engineers' opinions on the general functionality of this product. The engineers had a favourable outlook on the major feature of the model, with 73% believing that it will significantly enhance maintenance proactivity. This finding is highlighted in Table 7.4. In addition, 55% of respondents concur that this model can enhance engineers' comprehension of software systems, which was one of the study's objectives identified as lacking during the interview phase. It is important to mention that 8

out of 11 engineers had a neutral opinion on the probability of NS suppliers adopting this approach, as shown in Table 7.5. This is understandable given the complexity of predicting this instance, as NS has a diverse range of suppliers. This issue was posed with the anticipation that the primary users of this model would be the engineers responsible for designing the systems, in particular in the case of NS, because the suppliers are involved. Another noteworthy element is that the engineers who were polled had a good assessment of the model's effectiveness in mitigating the risk of incorrectly installed software or configuration. This issue was also identified as an ongoing concern during the interview process.

Answers	**Do you think that this model can help engineers better understand the given software and its functions?**	**Do you think that this model can help to improve the maintenance planning process?**	**Do you think that this model can have a positive effect on maintenance proactivity?**
Strongly disagree	0	1	1
Disagree	1	1	1
Neutral	2	4	1
Agree	6	5	8
Strongly agree	2	0	0

Table 7.4: Model features part 1

Answer	**Do you think that this model can help reduce the risk from wrongly installed software/configuration?**	**Do you think that the suppliers of NS will use this model?**
Strongly disagree	0	0
Disagree	1	2
Neutral	4	8
Agree	6	1
Strongly agree	0	0

Table 7.5: Model features part 2

The questionnaire also looked into the model's ability to accurately replicate possible failures and the potential benefits of using this tool in NS, as shown in Table 7.6. The majority opinions were overwhelmingly favourable, indicating that a prospective investment in this instrument may yield significant advantages for the organisation. According to the engineers' thoughts, using this model can provide several improvements

to the overall availability of the train, as shown in Table 7.7. However, there is a clear division of perspectives when it comes to enhancing safety. This can be attributed to the fact that some of the individuals interviewed did not acknowledge the software systems as being crucial for safety.

How effectively does the model simulate possible software failures?	Count	How useful can the model be if it is implemented within NS?	Count
Completely not effective	0	Completely useless	0
Slightly effective	2	Slightly useful	1
Moderately effective	5	Moderately useful	5
Quite effective	4	Quite useful	5
Very effective	0	Very useful	0

Table 7.6: Overall usefulness of the model

What do you think will be the added value for the overall train safety if this model is implemented?	Count	What do you think will be the added value for the overall train availability if this model is implemented?	Count
No added value	2	No added value	0
The safety will be increased slightly	5	The availability will be increased slightly	4
The safety will be increased moderately	4	The availability will be increased moderately	5
The safety will be increased a lot	0	The availability will be increased a lot	2

Table 7.7: Added value of the model for the train safety and availability

The final multiple-choice question of the poll inquired whether the engineers believed there were comparable or identical models to this one. Approximately 55% of respondents indicated that the model is moderately similar to an existing one, as shown in Table 7.8. This can be considered a drawback of the model, as it does not offer anything groundbreaking or innovative. However, the intention is to make the model less different from existing ones, making it easier to understand and implement. This increases the likelihood of NS's suppliers adopting it. Whether this is perceived as a drawback of the model depends on one's perspective.

Do you think the model is too similar to others already existing ones in NS?	Count
Not very similar at all	1
Slightly similar	4
Moderately similar	6
Very similar	0
Identical	0

Table 7.8: Check for the similarity with other models

7.3 Partial Conclusion

This chapter revealed the findings of the survey conducted in NS to authenticate the proposed novel methodology and the Excel tool that steps on it and the gathered input on its potential use and further benefits. A concise study was conducted to substantiate the provided tables. In the upcoming chapter, a comprehensive analysis and discussion of this data will be conducted.

Chapter 8

Discussion

8.1 Introduction

This chapter discusses the responses gathered from participants regarding the adapted Reliability-Centered Maintenance (RCM) logic model, shown in the previous chapter. Also, an evaluation will made of the established goals and objectives of this study and its main findings and suggestions throughout this report. The emphasis will be on the theoretical and practical implications of the findings, the limitations of this research, and possible directions for future research. The primary goal is to provide a critical analysis and thorough summary of the collected data for the proposed ideas, offering an overall view of the research's contribution to the maintenance field.

8.2 Summary of Key Findings

This study aimed to provide a clear and comprehensive definition of Operational and Informational technologies (OT and IT) in order to eliminate any uncertainty, particularly in regards to the identification of IT. This was achieved by a comprehensive examination of existing literature and a subsequent interview phase conducted inside the Dutch Railways (NS). From an industry perspective, the derived definitions were carefully formulated to eliminate any ambiguities associated with both technologies and the concept of software failure. This clarity is intended to not only enhance the recognition of the significance of such failures but also to prevent their potential underestimation in subsequent analyses and evaluations. For academia, it presents another point of view on software failure which can trigger additional research in the field, to study what is the potential effect of hardware failure on software reliability. This analysis has shown that due to the increasing convergence of the two technologies, the most efficient approach for assessment will be to look at them solely as hardware and software, particularly from a maintenance standpoint. While the study may have provided a clear definition of OT and IT, it is important to consider that technology is constantly evolving and definitions may need to be updated accordingly.

The main suggestions of this study are that traditional Reliability-Centered Maintenance methodologies, primarily used for hardware systems, can be adapted for software systems by incorporating specific modifications. These include selecting the appropriate system level for assessment, understanding software functions, and integrating graphical modelling tools like UML and SysML to enhance the understanding and management of software failures. Then modelling them through Software Failure Mode and Effect Analysis (SFMEA) and choosing the right strategy to mitigate every failure by using the proposed logic tree. However, software systems and hardware systems have fundamental differences in how they operate and fail, making it difficult to directly apply traditional RCM methodologies to software without significant adjustments.

8.3 Theoretical Implications

The primary objective of this research was to enhance the maintenance field by focusing on the opportunities presented by evolving technology, particularly software advancements. The absence of a systematic evaluation procedure for software failures has been identified as a deficiency in both the academic field and NS. Particularly the absence of a well-defined and compact model that can be used in many scenarios

and different systems. This work seeks to address the existing knowledge gap by proposing a technique for evaluating and mitigating software failures that can be implemented in many scenarios. While focusing on software failures may enhance maintenance practices, it is important to consider that the lack of a systematic evaluation procedure may not necessarily be the only deficiency in the field.

8.4 Practical Implications

The Practical implications of this study include the creation of an Excel model that applies the proposed principles for SRCM and demonstrates how these ideas can be practically implemented. The engineers interviewed in the course of this research within NS were provided with an explanatory video about the developed tool and a questionnaire to fill out to give their feedback. Although not statistically relevant, it still presented enough points for this discussion and showed diverse opinions from professionals about the real implementation of the tool and the suggested ideas. While the feedback from engineers within NS may provide valuable insights, relying solely on a small sample size within one organization may limit the general ability of the findings to other contexts. Additionally, self-reported feedback from professionals may not always accurately reflect their true opinions or behaviours.

8.5 Interpretation of Results

The survey with multi-choice questions from the participants highlights a generally positive reception towards using an adapted RCM logic model for modelling software failures, with a significant appreciation for its potential benefits in improving maintenance proactivity and train availability and its overall usefulness if it is implemented within NS. Also, a generally positive view was made in terms of the capability of the model to help with the understanding of the main functions of the software. However, there are clear indications of areas for improvement, particularly in enhancing the model's structure and usability, and quite a neutral position about the chance for NS's supplier to use it.

As the last question was left open, the surveyed employees could give their additional remarks on the new tool, shown in raw format in the appendices. The feedback reveals a mix of appreciation and concern, highlighting areas for potential improvement. Several respondents expressed confusion about the intended audience for the tool, questioning whether it is meant for NS or their suppliers, and at which stages—design or service—the tool should be applied. This ambiguity suggests a need for clearer communication and documentation. Additionally, some other users highlighted vulnerabilities in the presentation of the model and the survey itself as they found it difficult to follow, particularly the initial slides and the lack of a "don't know" option in the multiple-choice questions, which should be addressed in future iterations of the tool and particularly its instructional materials.

A significant portion of the feedback emphasised the tool's applicability at the early stages of software projects, where it can be integrated with existing risk mitigation methodologies within NS. However, there are reservations about its fit for Original Equipment Manufacturers (OEMs) and the challenge of engaging suppliers in using the model, given the proprietary nature of software source code and varying levels of expertise in software development environments. Respondents also pointed out the need to differentiate between hardware-related issues and actual software failures, advocating for a more nuanced approach that considers the distinct requirements for safety and security updates. This clearly showed one of the limitations of this study, the lack of proprietary testing with different software systems so that as many software failures could be modelled to showcase the potential of the idea and the model. That is a case that has to be addressed in the future development of both the SRCM process and the Excel tool.

Moreover, more concerns were raised regarding the practical implementation of the tool, particularly the involvement of suppliers. The feedback indicates that while the model is valuable for identifying risk and maintenance activities, NS might lack the in-house expertise to perform comprehensive analyses without collaboration with its suppliers. Ensuring that they are willing and able to use the tool, possibly through contractual obligations or financial incentives, could be crucial for its success. Additionally, understanding the suppliers' development environments, including documentation and versioning practices, is essential for accurately assessing risks and implementing effective Reliability-Centered maintenance for software. That

highlighted the challenges that such a tool can face when implemented and the other main limitation of this study, which was the inability to widen the process and include, for example, a supplier company of NS in the research. Therefore, addressing these points and limitations could be vital for the model's usability and effectiveness, ultimately contributing to better software reliability and maintenance outcomes, especially for the environment in which NS is working.

The participants' constructive input provides vital advice for future versions of the model, with the goal of maximising its effectiveness and value inside the company. The analysis emphasised the strengths and weaknesses of this research, enhancing its understanding of how software failure can be mitigated. It also identified areas that require further improvement and iterative development to fully realise the potential of this promising idea and tool.

8.6 Limitations

This study faced several limitations that should be acknowledged. Firstly, the focus was primarily on software failures within the context of the Dutch Railways (NS), which may limit the general applicability of the findings to other organisations or industries. The sample size for feedback was relatively small and confined to NS, potentially impacting the robustness and external validity of the results. Additionally, the study relied on self-reported feedback from professionals, which may not always accurately reflect their true opinions or behaviours due to biases or misunderstandings. Another significant limitation was the lack of proprietary testing with different software systems, restricting the ability to showcase the model's potential fully. Also, proposing a technique for evaluating software failures may not fully address all potential challenges emerging for the new technologies. Moreover, focusing solely on hardware and software may overlook the importance of other factors such as cybersecurity issues in assessing OT and IT systems. Finally, the complexity and dynamic nature of software systems may require further adaptation and a more flexible approach than what is proposed in this study for the Software Reliability-Centered Maintenance (SRCM) methodologies.

8.7 Partial Conclusion

The chapter presented a thorough examination and evaluation of the primary results of this study and the collected feedback on the generated model, which is based on the modified Reliability-Centered Maintenance (RCM) model for software failures. The study's primary findings were emphasised, highlighting the potential of the suggested new concepts and methodologies, as well as providing a critical perspective to identify shortcomings and gaps within them.

Chapter 9

Conclusion

9.1 Summary of the Research

Today's systems in the railway industry rely more and more on transferring information with one another for safe and effective operations, thus bridging the gap between informational and operational technology ever so closely together. This means that software will play an even bigger role than before in the operations of these converged systems. Thus, failures originating from badly designed or maintained software are becoming a big challenge for the industry. This research identified these challenges by researching the understanding of both technologies and their convergence. That formed the objectives set to be achieved to understand what are the current preventive methods to mitigate the aforementioned challenge, what are their gaps, and how to bridge them:

1. Identify the main way of understanding the nature and failure mechanisms for IT systems and the methods of prediction of their key software and hardware failures in the rolling stock;
2. Develop a method to incorporate the effects of software failures into IT-OT converged systems within the RAMS process;
3. Demonstrate the added value of incorporating the effects of software failures concerning system performance.

To achieve the first objective, extensive research was done within the Dutch Railways (NS). The current RAMS methods for assessing failures were analysed, and fifteen people were interviewed to research their experiences and opinions with IT and OT systems, a particular emphasis was placed on how they are dealing with software failures. This highlighted a gap, specifically in the method used by NS, about a process that focuses on assessing and mitigating failures originating from software systems.

9.2 The viewpoint for IT/OT converged systems

This study provides a thorough examination of Operational Technology (OT) and Information Technology (IT), emphasizing their convergence and significance in the railway sector. Three important conclusions was made out of this research about IT and OT converged systems:

- The convergence of Operational Technology (OT) and Information Technology (IT) into Cyber-Physical Systems (CPS) is transforming the railway industry.
- Conceptualizing IT and OT as hardware and software components facilitates accurate failure assessments and bridges the gap between software and mechanical engineers.
- Addressing the challenges posed by new technology-induced software failures requires new investigative and testing approaches, forming the foundation for the next step of the research.

9.3 Role of RCM in Addressing Software Failures

The second objective was achieved by conducting a literature review of the available RAMS methods that deal with software failures, which resulted in the finding that Software Failure Mode and Effect Analysis(SFMEA) was one of the most effective methods to model and assess such failures. With the background knowledge that FMEA is part of a bigger method called Reliability-Centered Maintenance, a gap in academia was found in the research on how RCM can be adapted so software failures can fit within it. Reliability-Centered Maintenance (RCM) has been identified as a viable method to model and manage software failures within IT-OT systems. RCM's structured approach focuses on identifying potential failure modes, assessing their impacts, and implementing maintenance strategies to mitigate risks. This method, traditionally used in physical asset management, could prove effective in the digital domain for several reasons:

- Systematic Analysis: RCM provides a systematic framework for analysing software failure modes, enabling a comprehensive understanding of potential issues.
- Proactive Maintenance: By predicting and preventing failures before they occur, RCM enhances system reliability and reduces unplanned downtime.
- Cost Efficiency: Implementing RCM can lead to significant cost savings by optimising maintenance activities and extending the lifespan of IT-OT assets.

9.4 Developed Excel tool for implementation

The third objective was to demonstrate the added value of assessing software failure regarding system performance and this was achieved by developing an Excel-based tool and putting it under evaluation from a variety of experts from NS to demonstrate how RCM principles can be applied to manage software failures in IT-OT converged systems. This tool provides a practical means to:

- Model Software I/O, Functions, and Failure Modes: Users can input the main software functions into the given system component and, upon this to model respective failure modes, facilitating a thorough analysis.
- Risk Assessment: The tool enables the assessment of failure impacts and causes, helping prioritise maintenance efforts based on risk levels.
- Maintenance Planning: By following the RCM logic and providing the user with an RCM logic tree, the most effective maintenance strategy to mitigate the given software failure can be selected, which aids in enhancing the efficient planning and execution of maintenance activities.

9.5 Overall potential of SRCM

According to feedback from engineers with different backgrounds in NS, the idea of using Software RCM methods in the processes for mitigating software failures has a lot of benefits and potential. This could be valuable for industries that want to make their software systems more reliable and efficient. The developed Excel tool exemplifies how theoretical concepts can be translated into practical applications, providing a roadmap for organisations to follow. Key implications include:

- Enhanced Decision-Making: With a clear understanding of potential software failure modes and their impacts, organisations can make informed decisions about maintenance and upgrades.
- Improved Software System Reliability: Proactive maintenance strategies reduce the likelihood of software system failures, leading to more stable and reliable operations.
- Resource Optimisation: Efficient maintenance planning ensures that resources are allocated effectively, minimising waste and reducing operational costs.

9.6 Limitations and Future work

Several limitations of this study should be outlined, such as:

- The lack of comprehensive testing of the Excel model in a real project.
- Conducting a restricted range of tests on the model using various systems to demonstrate further advantages and disadvantages of the concept and the model.
- The limited timeframe to further research the other possibilities to use assistive techniques, such as Software Fault Tree Analysis (SFTA) to be used within the SRCM logic and allow for a root cause analysis of every software failure, thus enhancing the choice of implied maintenance strategy.

While this research provides valuable insights into the challenges of IT-OT convergence and their software failure by proposing new methods to deal with them, several areas warrant further exploration:

- SFTA within SRCM: Further research is needed to check how other assistive techniques, such as SFTA could be also integrated into the SRCM logic.
- Cybersecurity Integration: Further research is needed to develop comprehensive frameworks that address both operational reliability and cybersecurity within IT-OT systems.
- Scalability and Adaptability: Exploring ways to scale the developed Excel tool for larger and more complex systems can broaden its applicability. Also, further 'field' tests are needed to prove its right functioning and improve it.
- Advanced Analytics: Integrating advanced analytics and machine learning with RCM could enhance predictive maintenance capabilities, specifically for software failures, as they tend to be harder to predict.

9.7 Final Thoughts

In conclusion, the convergence of IT and OT systems is a double-edged sword, offering significant advantages alongside considerable challenges. The integration of digital technology will have a significant impact on the daily functioning of the rolling stock, resulting in numerous software failures that designers and operators will need to address. By adopting RCM methods to manage these failures, industries can mitigate risks and enhance system reliability. However, selecting the appropriate maintenance strategy is a crucial aspect of this process and can be influenced by various factors. One factor to consider is the Risk Priority Number (RPN), which is assigned a score for each failure. Another factor is the criticality of each failure and its impact on the overall operation of the train or the machine. This decision is influenced by both the software designer and the operating company, the smooth communication and clear opinions of these two parties play a crucial role in determining the outcome of the SRCM logic tree and thus the effectiveness of mitigating every software failure. Ongoing research and innovation will be imperative for fully realizing the potential of the proposed Software Reliability-Centered Maintenance process, particularly in ensuring the sustained operational integrity of integrated software systems.

References

Awad Ali, Mohammed Bakri Bashir, Alzubair Hassan, Rafik Hamza, Samar M. Alqhtani, Tawfeeg Mohmmed Tawfeeg, and Adil Yousif. Design-time reliability prediction model for component-based software systems. *Sensors*, 22, 4 2022. ISSN 14248220. doi: 10.3390/s22072812.

Uchenna P. Daniel Ani, Hongmei (Mary) He, and Ashutosh Tiwari. Review of cybersecurity issues in industrial critical infrastructure: manufacturing in perspective. *Journal of Cyber Security Technology*, 1:32–74, 1 2017. ISSN 2374-2917. doi: 10.1080/23742917.2016.1252211.

Chrysostomos Symvoulidis Athanasios Kiourtis and Dimosthenis Kyriazis Argyro Mavrogiorgou. *Capturing the reliability of unknown devices in IoT world.* 2018. ISBN 9781538695852.

ATSB. In-flight upset - 154 km west of learmonth, WA, 7 October 2008, VH-QPA Airbus A330-303.

Ernnie Illyani Basri, Izatul Hamimi Abdul Razak, Hasnida Ab-Samat, and Shahrul Kamaruddin. Preventive maintenance (pm) planning: a review. *Journal of quality in maintenance engineering*, 23(2):114–143, 2017.

Ron Bell. Introduction to iec 61508. In *Proceedings of the 10th Australian workshop on Safety critical systems and software-Volume 55*, pages 3–12. Citeseer, 2006.

Carlson. Understanding the fundamental definitions and concepts of fmeas failure mode and effects analysis (fmea). 2012.

Jesús Carretero, José M. Pérez, Félix García-Carballeira, Alejandro Calderón, Javier Fernández, Jose D. García, Antonio Lozano, Luis Cardona, Norberto Cotaina, and Pierre Prete. Applying rcm in large scale systems: A case study with railway networks. *Reliability Engineering and System Safety*, 82:257–273, 12 2003. ISSN 09518320. doi: 10.1016/S0951-8320(03)00167-4.

Ana Paula Henriques de Gusmão, Maisa Mendonça Silva, Thiago Poleto, Lúcio Camara e Silva, and Ana Paula Cabral Seixas Costa. Cybersecurity risk analysis model using fault tree analysis and fuzzy decision theory. *International Journal of Information Management*, 43:248–260, 12 2018. ISSN 02684012. doi: 10.1016/j.ijinfomgt.2018.08.008.

Todd Dewett and Gareth R Jones. The role of information technology in the organization: a review, model, and assessment. *Journal of management*, 27(3):313–346, 2001.

Fateme Dinmohammadi, Babakalli Alkali, Mahmood Shafiee, Christophe Bérenguer, and Ashraf Labib. Risk evaluation of railway rolling stock failures using fmeca technique: A case study of passenger door system. *Urban Rail Transit*, 2:128–145, 12 2016. ISSN 21996679. doi: 10.1007/s40864-016-0043-z.

EN 50128:2011. Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems. Standard, CENELEC, Avenue Marnix 17, B - 1000 Brussels, July 2011.

EN 50129:2018. Railway applications - Communication, signalling and processing systems - Safety related electronic systems for signalling. Standard, CENELEC, Avenue Marnix 17, B - 1000 Brussels, December 2018.

EN 50716:2023. Railway Applications - Requirements for software development. Standard, CENELEC, Avenue Marnix 17, B - 1000 Brussels, December 2023.

Emmanuel Francalanza, Mark Mercieca, and Alec Fenech. Modular system design approach for cyber physical production systems. volume 72, pages 486–491. Elsevier B.V., 2018. doi: 10.1016/j.procir.2018.03.090.

Gartner. Information technology glossary, 2021.

Y González-Arechavala, J A Rodríguez-Mondéjar, and G Latorre-Lario. The opportunity to improve software rams. *WIT Transactions on State of the Art in Science and Engineering*, 46:1755–8336, 2010. doi: 10.2495/978-1-84564. URL www.witpress.com,.

Peter Hehenberger, Thomas J. Howard, and Jonas Torry-Smith. *From mechatronic systems to cyber-physical systems: Demands for a new design methodology?*, pages 147–163. Springer International Publishing, 1 2016. ISBN 9783319321561. doi: 10.1007/978-3-319-32156-1_10.

IEC 62628:2012. Guidance on software aspects of dependability. Standard, CENELEC, Avenue Marnix 17, B - 1000 Brussels, September 2012.

Anne Immonen and Eila Niemelä. Survey of reliability and availability prediction methods from the viewpoint of software architecture. volume 7, pages 49–65, 2 2008. doi: 10.1007/s10270-006-0040-x.

H Islam et al. Reliability-centered maintenance methodology and application: a case study. *Engineering*, 2010, 2010.

Capers Jones. *Software assessments, benchmarks, and best practices*. Addison-Wesley Longman Publishing Co., Inc., 2000.

Dale W Jorgenson and Kevin J Stiroh. Information technology and growth. *American Economic Review*, 89(2): 109–115, 1999.

S Z Kamal, S M Al Mubarak, Saudi Aramco, B D Scodova, BHP Billiton, P Naik, P Flichy, Baker Hughes, and G Coffin. Spe-181087-ms it and ot convergence-opportunities and challenges, 2016.

Arno Kok, Alberto Martinetti, and Jan Braaksma. Rams never dies! applying the approach to it/ot converged systems. pages 3181–3187. Research Publishing Services, 2023. ISBN 978-981-18-8071-1. doi: 10.3850/978-981-18-8071-1_P286-cd. URL https://www.rpsonline.com.sg/proceedings/esrel2023/html/P286.html.

M Kraeling and D Fletcher. Railroad assets: information and operational (it/ot) convergence. In *AusRAIL PLUS 2017, Rail's Digital Revolution, 21-23 November 2017, Brisbane, Qld, Australia*, 2017.

Abhishek Kumar, Puneet Sharma, and Amit Agarwal. Rams of hvac for rolling stock application. *International Journal of Engineering Research and Technology (IJERT) www.ijert.org*, 10, 2021. ISSN 2278-0181. URL www.ijert.org. Calculation about failure rates
Also parts that are controlled by software are included - there failure rates are calculated.
.

Tung La-Ngoc and Gihwon Kwon. Comparing the effectiveness of sfmea and stpa in software-intensive railway level crossing system. volume 474, pages 1281–1288. Springer Verlag, 2018. ISBN 9789811076046. doi: 10.1007/978-981-10-7605-3_204.

Jay Lee and Behrad Bagheri. *Cyber-physical systems in future maintenance*, volume 20, pages 299–305. Springer Heidelberg, 2015. ISBN 9783319155357. doi: 10.1007/978-3-319-15536-4_25.

Zhang Hong Xu Lili. *An Improved SFMEA Method Integrated with Assistive Techniques*. IEEE, 2013. ISBN 9781479925520.

Qamar Mahboob and Enrico Zio. *Handbook of RAMS in railway systems: theory and practice*. CRC Press, 2018.

Sipke Van Manen, Ed Brandt, Jaap Van Ekris, and Wouter Geurts. Topaas: An alternative approach to software reliability quantification. *Quality and Reliability Engineering International*, 31:183–191, 3 2015. ISSN 10991638. doi: 10.1002/qre.1570.

Fausto Pedro García Márquez, Jesús María Pinar Pérez, Alberto Pliego Marugán, and Mayorkinos Papaelias. Identification of critical components of wind turbines using fta over the time. *Renewable Energy*, 87: 869–883, 2016.

Ben Swarup Medikonda and P. Seetha Ramaiah. Software safety analysis to identify critical software faults in software-controlled safety-critical systems. volume 249 VOLUME II, pages 455–465. Springer Verlag, 2014. ISBN 9783319030944. doi: 10.1007/978-3-319-03095-1_48.

M. A. De Miguel, J. F. Briones, J. P. Silva, and A. Alonso. Integration of safety analysis in model-driven software development. *IET Software*, 2:260–280, 2008. ISSN 17518806. doi: 10.1049/iet-sen:20070050.

László Monostori. Cyber-physical production systems: Roots, expectations and research and development challenges. volume 17, pages 9–13. Elsevier B.V., 2014. doi: 10.1016/j.procir.2014.03.115.

John Moubray. *Reliability-centered maintenance*. Industrial Press Inc., 2001.

MA Muhammed Nor, AF Yusop, MA Hamidi, MN Omar, NA Abdul Hamid, and WM Wan Mohamed. Alternative railway tools and sustainability in rams: A review. In *International Conference on Mechanical Engineering Research*, pages 541–554. Springer, 2021.

G Murray, M N Johnstone, and C Valli. The convergence of it and ot in critical infrastructure. pages 149–155, 2017. doi: 10.4225/75/5a84f7b595b4e. URL https://ro.ecu.edu.au/ismDOI:https://doi.org/10.4225/75/5a84f7b595b4e.

MAINTENANCE NASA GUIDE. Rcm guide. 2008.

David J Parker and Yiannis I Papadopoulos. Optimisation of networked control systems using model-based safety analysis techniques, 2007.

Dewanne M. Phillips, Thomas A. Mazzuchi, and Shahram Sarkani. An architecture, system engineering, and acquisition approach for space system software resiliency. *Information and Software Technology*, 94: 150–164, 2 2018. ISSN 09505849. doi: 10.1016/j.infsof.2017.10.006.

Jim V Picknell. Is rcm the right tool for you? *Plant Engineering and Maintenance*, 23(6):23–36, 1999.

Janusz Poliński and Krzysztof Ochociński. Digitization in rail transport. *Problemy Kolejnictwa - Railway Reports*, 64:137–148, 9 2020. doi: 10.36137/1885e.

Ralf Reussner, Michael Goedicke, Wilhelm Hasselbring, Birgit Vogel-Heuser, Jan Keim, and Lukas Märtin. *Managed software evolution*. Springer Nature, 2019.

N H Roberts and D F Haasl. Fault tree handbook, 1981.

Atiq Waliullah Siddiqui and Mohamed Ben-Daya. Reliability centered maintenance. In *Handbook of maintenance management and engineering*, pages 397–415. Springer, 2009.

Jonathan Sillito and Esdras Kutomi. Failures and fixes: A study of software system incident response. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 185–195. IEEE, 2020.

Keith Stouffer, Michael Pease, CheeYee Tang, Timothy Zimmerman, Victoria Pillitteri, Suzanne Lightman, Adam Hahn, Stephanie Saravia, Aslam Sherule, and Michael Thompson. Guide to operational technology (ot) security. Technical Report NIST SP 800-82 Rev. 3, National Institute of Standards and Technology (NIST), 2023.

E Burton Swanson. The dimensions of maintenance. In *Proceedings of the 2nd international conference on Software engineering*, pages 492–497, 1976.

Hussein David Tafur, Giacomo Barbieri, and Carlos Eduardo Pereira. An fmea-based methodology for the development of control software reliable to hardware failures. volume 54, pages 420–425. Elsevier B.V., 2021. doi: 10.1016/j.ifacol.2021.08.047.

Massood Towhidnejad, Dolores R Wallace, and Albert M Gallo. Validation of object oriented software design with fault tree analysis. In *28th Annual NASA Goddard Software Engineering Workshop, 2003. Proceedings.*, pages 209–215. IEEE, 2003.

Gregory Travis. How the Boeing 737 Max Disaster Looks to a Software Developer — spectrum.ieee.org. https://spectrum.ieee.org/how-the-boeing-737-max-disaster-looks-to-a-software-developer, 2024. [Accessed 28-05-2024].

Alan C Tribble. Software safety analysis of a flight guidance system, 2002.

Ronald L Wasserstein and Nicole A Lazar. The asa statement on p-values: context, process, and purpose, 2016.

WhatisSysML2023. What is sysml - visual paradigm guides. URL https://guides.visual-paradigm.com/what-is-sysml/.

Wikipedia-UML. Unified Modeling Language - Wikipedia — en.wikipedia.org. https://en.wikipedia.org/wiki/Unified_Modeling_Language. [Accessed 29-05-2024].

Zhe Zhang, Limin Jia, and Yong Qin. Rams analysis of railway network: model development and a case study in china. *Smart and Resilient Transportation*, 3:2–11, 5 2021. ISSN 2632-0487. doi: 10.1108/srt-10-2020-0013. Safety analisys into train accidents
.

Chapter 10

Appendices

10.1 Disclosure

During the preparation of this work, I used *'QuillBot AI'* and *'Grammarly AI'* to enhance the quality of writing in this report, I also used *'ChatGPT'* for the data processing that was done during this study. After using those tools/services, I thoroughly reviewed and edited the content as needed, taking full responsibility for the final outcome.

10.2 Validation - Additional comments

The last question of the survey was an open one, so the engineers could give their open opinions, suggestions or critiques. 8 of the engineers gave their additional points of view on different aspects, which also made the comments hard to combine, so they are presented below in 4 parts as screenshots.

15. If you have any additional remarks you can leave them below.

8 Answer

ID ↑	Name	Answer
1	Anonymous	The audience of the new tool is not clear for me. Will it be used by NS or suppliers? Is it meant to use during the design phase or during service or both? What type of SFMEA's do suppliers use at the moment? For the multiple choice answers, I don't always have a good understanding. For next time, a multiple choice answer 'don't know' would be a good add on. In the movie it is a bit hard to follow both the spoken language and the sheets. Especially the first 2 or 3 sheets are difficult to follow.
2	Anonymous	I think this model is very useful in the beginning of a SW project. Not for SW that is already implemented. On the other hand, there are other methods for risk mitigation on software development in an early stage also already proven in methodology. So my opinion is this model is use full if you want to fit it in currently known models within NS for software projects within NS. I don't think it fits for OEM'ers.

Figure 10.1: Survey comments - Part 1

3	Anonymous	I wanted to answer 'not applicable' for Q10-11-12 since I feel uninformed to make such a formulated opinion. The model proves a valuable method to make risk assessment and identify corrective/preventive maintenance activities. However, I still have some doubts if NS has all the available knowledge inhouse in order to make such an analysis. The supplier plays an important role in the reliability analysis of random/systematic software failure (often expressed using RAMS criteria in the contract). It would be a challenge to seduce suppliers to use this model - I feel they wont use it as long as NS does not require this (financial agreements etc.) Nevertheless, it's a very useful framework and is able to spot areas of improvements NS needs to take to get more control in reliability centred maintenance for software!
4	Anonymous	The examples show a lot of hardware failures with the related handling of the software. Not actual software failures, ie. Kernel Panic, Process crash, currupted files. etc. Therfore for me these are hardware failures (with the related handling of these errors). But this might be a mismatch in my perception. Also, in my opinion, you cannot combine Safety and Security into one group. Safety system have a much larger homologation spec. Wheras security updates, for example on the OBIS platform, has very limited (to none) homologation needs.

Figure 10.2: Survey comments - Part 2

5	Anonymous	- Question is if Supplier want to use RCM-SFMEA (contractual/CAPEX) - For the questions in relation to Safety, how is SIL taken into account?
6	Anonymous	I think the toughest part is to let the suppliers use the model. For a maintenance engineer at NS, it is almost impossible to fill in this sheet because of lack of knowledge about the sourcecode. If suppliers are open to fill in the model, I think it can be very useful in understanding the possible risks and probabilities related to RAMS. Some topics to think of: - The current model seems a system/component related. But most systems have a lot of interfaces to other systems, especially TCMS. It really depends on the input and output from other systems. Will these kind of systems also work with the model? - What about the transport layers. We often see problems within the transport networks like CAN, MVB. This can be either hardware or software problems. Will the tool help to identify those issues too? - I think one of the most important aspects with software programming is the software development environment at the supplier. How do they deal with documentation, versioning, issue tracking, testing. Sometimes we see software which looks like it is made by an engineer with some limited software skills. And sometimes we see software from suppliers that have huge software teams with clear versioning and good documentation. I think to have a good understanding about the risks and severity it is also important to know more about the suppliers, organisation/environment and professionalism related to software development.

Figure 10.3: Survey comments - Part 3

7	Anonymous	For me, it's important that the people involved are instructed in how to use the model and the consequences. (I expect as not being a RAM engineer it is understandable. In video slide25/51 has a syntax error. (function)
8	Anonymous	The model reacts on given input from de software definition proces. Or how the software should work. In most of the cases, this is intellectual property of the supplier. in my opinion the model does not work for real software failures.

Figure 10.4: Survey comments - Part 4