TU WIEN Informatics

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Semi-Automatic Generation of Production Exception Handling for Agile Process Guidance

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Hugo Blaess
Matrikelnummer 12302490

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Stefan Biffl
Mitwirkung: Univ.Lektor Dipl.-Ing. Dr.techn. Dietmar Winkler
Dipl.-Ing. Sebastian Kropatschek
Prof. Dr. Ing. HDR Frédérique Laforest

Wien, 1. September 2025

Hugo Blaess | Stefan Biffl

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Semi-Automatic Generation of Production Exception Handling for Agile Process Guidance

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Hugo Blaess
Registration Number 12302490

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Stefan Biffl
Assistance: Univ.Lektor Dipl.-Ing. Dr.techn. Dietmar Winkler
Dipl.-Ing. Sebastian Kropatschek
Prof. Dr. Ing. HDR Frédérique Laforest

Vienna, September 1, 2025

Hugo Blaess | Stefan Biffl

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Hugo Blaess

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 1. September 2025

Hugo Blaess

Acknowledgements

First acknowledgment goes to the Austrian Center for Digital Production (CDP). The CDP provided an inspiring research environment and valuable resources that shaped many aspects of this work. Their focus on innovative and applied digital production solutions strongly influenced the practical orientation of this thesis.

I want to start with a sincere thank you to Sebastian Kropatschek. He gave me huge help during this thesis. I asked a lot of questions and made mistakes along the way, but he always stayed patient and took time to explain things. Without his support, I honestly think I wouldn't have finished this. He gave me advice when I got stuck and reminded me to keep going even when it was hard. I'm really grateful for all the time and effort he gave me.

I also want to thank Univ.Lektor. Dr. Dietmar Winkler for helping me with the writing. Academic writing was not easy for me, and I know I still have a lot to learn, but his comments helped me understand how to improve. I learned a lot from that process.

Special mention for Univ.Prof. Dr. Stefan Biffl & Frédérique Laforest who supervised my work. The feedback helped me to keep a consistent focus and structure and allowed me to publish something I am proud of.

A special thanks to my fellow students Elias Moser and Tobias Gruber, who started this whole project, and to all the others who worked on it in parallel. In particular, I want to acknowledge Johannes Fesl and Andreas Fuchs for their collaborative spirit and valuable contributions throughout this project.

Thanks to my friends, classmates, and everyone I met during my studies here. Some days were difficult, and talking or studying together really helped. I also want to thank my family. Even though we were far apart, their support and belief in me kept me going.

And finally, thank you to all the teachers and staff who were part of my time here. I learned something from each course, and it all helped me get to this point.

Kurzfassung

Im Kontext moderner Produktionssysteme, die insbesondere im Automobilsektor mit wachsender Komplexität konfrontiert sind, bieten cyber-physische Produktionssysteme (CPPS) neue Möglichkeiten zur Effizienzsteigerung. Viele Fabriken verlassen sich bei Ausnahmen jedoch noch auf statische Handbücher oder Experteneingriffe, was die Reaktionszeit verlangsamt und die Aktualität der Verfahren erschwert.

Die größte Herausforderung ist das Fehlen agiler Echtzeit-Anleitungen für Bediener bei unerwarteten Ereignissen. Zwar existiert Wissen in Formaten wie FMEA-Berichten, ist aber vor Ort schwer nutzbar. Dies erschwert Schulung, erhöht die Expertenabhängigkeit und verringert die Autonomie der Bediener.

Ziel dieser Arbeit ist es, Expertenwissen in klare, schrittweise Prozesse zu überführen, die in Echtzeit über eine benutzerfreundliche Oberfläche verfügbar sind. Dadurch soll die Ausnahmebehandlung verbessert, die Fehlerquote reduziert und das kontinuierliche Lernen unterstützt werden.

Die Methode folgt dem Design Science Framework. Auf eine systematische Literaturrecherche und szenariobasierte Anforderungsanalyse folgte die Entwicklung eines Prototyps mit agiler, modularer Architektur. Das System nutzt einen Wissensgraphen mit FMEA-Daten, unterstützt Live-Bearbeitung und Versionskontrolle und wurde in einem industriellen Anwendungsfall über Interviews und Benutzerakzeptanzstudien evaluiert.

Die Ergebnisse zeigen, dass das APEH-System kontextbezogene Anleitungen bereitstellt, reparierbare und nicht reparierbare Ausnahmen behandelt und Prozesse mit Produktionsänderungen abgleicht. Die Benutzer fanden es nützlich und einfach zu bedienen, trotz einiger Einschränkungen in komplexen Situationen.

Zusammenfassend demonstriert der APEH-Prototyp die Machbarkeit und Vorteile halbautomatischer Prozessführung in CPPS. Er unterstützt Bediener, reduziert Abhängigkeit von statischer Dokumentation und zentralisiert Wissen für die Wiederverwendung. Zukünftige Arbeiten sollten sich auf Skalierbarkeit, Integration zusätzlicher Echtzeitdaten und die Verfeinerung der Benutzerinteraktion konzentrieren.

Abstract

In the context of modern manufacturing where production systems face more complexity, especially in the automotive sector, Cyber-Physical Production Systems (CPPSs) bring new capabilities that should help the industry face today's efficiency challenges. When considering exception handling, many factories still rely on static manuals or expert intervention during production exceptions. This slows down response time and creates difficulties in keeping procedures up to date.

The main challenge is the lack of agile, real-time guidance for operators during unusual or unexpected events. Knowledge exists in formats like FMEA reports, but is not always convenient to use on the site. This makes training harder, increases reliance on experts, and reduces the autonomy of operators.

The objective of this work is to transform expert knowledge into clear, step-by-step processes, available in real time through a user-friendly interface. The aim is to improve exception handling, reduce defect rates, and support continuous learning for staff.

The method follows the Design Science framework. First, a systematic literature review and a scenario-based requirements analysis were conducted to understand the needs of stakeholders and the capabilities required. Then, a prototype was developed using an agile, modular architecture. The system uses a knowledge graph fed by FMEA data, supports live editing and version control. Finally, the prototype was evaluated in an industrial use case through semi-structured interviews and user acceptance studies.

Results show that our APEH system can provide contextual guidance, handle repairable and unrepairable exceptions, and keep processes coherent with production changes. Users found it useful and easy to use, although some limitations remain when dealing with specific situations.

In conclusion, the prototype demonstrates the feasibility and benefits of semi-automatic agile process guidance in CPPS. It supports operators, reduces dependence on static documentation, and centralizes knowledge for reuse. Future improvements should focus on scaling the system, integrating more real-time data, and refining user interaction for complex cases.

Contents

CHAPTER 1

Introduction

1.1 Context and Motivation

In this section, we first go through the context for this work and then mention our main motivations

1.1.1 Context

In a world that is mainly built on services, manufacturing represented only 15% of the world's GDP in 2023[1]. Since the Industrial Revolution, and still today, manufacturing plays a crucial role in daily life and economic development. Almost every product goes through industrial manufacturing processes. To survive in an ever more competitive world, manufacturers have increasingly adopted advanced technologies to improve efficiency and meet new customer requirements [4].

This is particularly evident in the automotive industry, which is one of the most complex and competitive sectors. The industry is under constant pressure to meet high-quality standards, maintain safety regulations and reduce costs. These challenges include global supply chain disruptions, short product development cycles, and increased product customization [68]. They also face new sustainability issues where the rapid development of low-emission vehicles has become a priority that increases production complexity.

To address this increasing complexity, performance improvement is essential to ensure competitiveness and profitability. One major objective is to reduce defect rates that cause higher production times, increased costs, and reduced customer satisfaction. Preventive strategies and fast response systems are therefore needed to handle production exceptions and avoid costly disruptions [59].

[1] World Bank Group, *Manufacturing, value added (% of GDP)*, 2023. Available at: `https://data.worldbank.org/indicator/NV.IND.MANF.ZS` [Accessed: 2025-06-27].

A domain that directly serves this cause is quality assurance (QA). According to Illes (2017) [31], a robust QA strategy is fundamental for continuous improvement and customer satisfaction. With this, we want to ensure that products meet customer requirements and regulatory standards. Within a sector transitioning toward more agile and digital processes, QA must adapt to faster and more flexible production lines. It involves both preventive measures (e.g., design reviews, process planning) and inspection activities (e.g., final tests, audits). Automation is a way to address these challenges, but human control and domain knowledge remain essential.

One commonly used framework for QA in the automotive industry is Failure Mode and Effects Analysis (FMEA). It is a systematic method used to identify potential failures in a process or product and assess their impact, it helps to prevent faults before they reach the customer. This framework is so powerful that it is required by some standards of the automotive industry, such as IATF 16949. It is often conducted manually, which can be time-consuming and error-prone, but automated or semi-automated tools for FMEA documentation are gaining interest, especially with access to operational data [61].

This transition to a data aware analysis is made possible by the rise of what we call the Industry 4.0. This refers to the integration of cyber-physical systems, the Internet of Things (IoT), and smart automation into traditional manufacturing processes [30]. This new industrial paradigm changes how factories operate. It leverages communication between machines, monitoring, and process adaptation based on real-time data. For manufacturers, this means better flexibility, improved productivity, and a greater possibility for customization. We can also see it as a digitization of industry plants where information is converted to digital format allowing more automation and intelligent decision-making [66].

1.1.2 Motivation

Within this context of exception handling in automotive CPPS, several issues can be identified that motivate this work.

Although CPPS are evolving rapidly towards something more efficient and reliable, these developments remain primarily focused on increasing production rates and shortening production times. This often leads to default rates being overlooked or even worsened. In this context, recurring failures and production exceptions persist. Existing strategies (such as FMEA) focus on identifying and understanding the cause of these problems, but there is still a gap in the area of prevention and reparation [19].

The consequences of this gap are evident in frequently high defect rates, reduced productivity, and losses in terms of quality. These represent critical issues for a production plant as they cause direct costs due to rework resources and indirect costs due to delays, sales lost, potential penalties. A loss of quality can also create customer dissatisfaction and damage the brand image.

We also observe that the currently used methods lack efficiency. Even with the rise of Industry 4.0 and digitization, we observe that manual documentation and paper-

based troubleshooting approaches remain widespread, despite being outdated and proven inefficient. Those methods are slow to adapt, hard to maintain, and not scalable [63].

With current approaches, we also observe some challenges in relation to knowledge transfer and operational training. While experts work together with FMEA to analyze default situations, it turns out that this valuable expert knowledge often remains confined to FMEA reports or spreadsheets. Those are sometimes lost in an online file systems, printed and compiled into large workbooks. Currently, there is no systematic way to convert this knowledge into usable process workflows. The same issue occurs regarding the training of operators, with data lost in exhaustive and non-intuitive manuals, transferring expert knowledge to less experienced staff remains difficult. This leads to high training effort with low efficiency.

In general, regarding trained operators on the production sites, a lack of autonomy is observed, with operators relying heavily on expert intervention. For their daily business they barely need any help, but when it comes to unexpected and unusual exception handling, we are missing a structured, step-by-step guidance support, since no one can be expected to excel at extraordinary tasks. In addition, operators often lack immediate insight into system failures, hindering fast decision-making and increasing impacts [8].

1.2 Use Case and Challenges

This work being done together with a use-case, we will briefly describe it before enumerating our main stakeholders and the challenges they might face.

1.2.1 Use Case

To gain a deeper understanding of the topic, we focus on a defined use case. This helps to better understand the domain, gain insights into the problem, and evaluate the solution in a real-world environment.

We collaborate with an Austria-based key supplier in the automotive industry, specialized in high-precision aluminum components. It operates production lines with strict quality and performance requirements to serve clients in the premium vehicle segment. The produced parts are safety-critical and must meet high quality standards, increasing the need for robust exception handling, especially as complexity and tight tolerances amplify the impact of even minor process deviations.

As shown in Figure 1.1, when facing a welding issue, the company relies on static documentation based on the experience of the workers. There is a clear need for flexible guidance that is context- and role-specific for employees.

The company already has processes in place to meet standard requirements. To perform FMEA as part of their quality and risk management, they use APIS IQ-RM [2] which

[2]APIS Informationstechnologien, *APIS IQ-Software Version 8.0*, 2025. Available at: `https://www.apis.de/en/software/products/apis-iq-software-version-8-0` [Accessed: 2025-07-01].

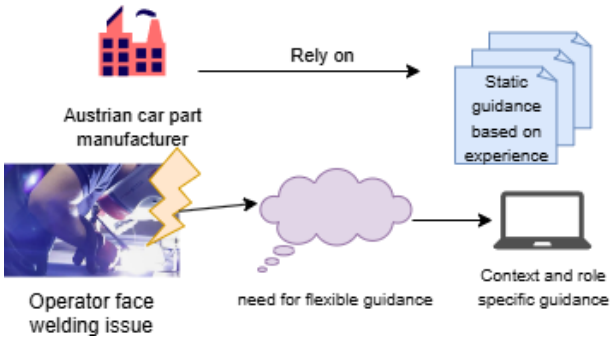


Figure 1.1: Diagram summarizing the Use Case

is widely used in the automotive industry. APIS serves as a knowledge input source, enriching the FMEA data with structured cause–effect relations that inform which types of exceptions should be prioritized. This system currently lacks procedural output: it provides only a brief diagnosis but does not deliver a repair workflow or proactive instructions.

The company already uses an Expert System to report and analyze anomalies during production. It suggests probable root causes based on known patterns and rules. This system currently lacks procedural output, it only provides a brief diagnosis but does not provide a repair workflow or proactive instructions. It is also difficult to scale and to integrate with external software. This is typically the type of system that we want to replace.

Typically, the environment is ever-changing: machines are upgraded, tools wear out, products evolve, and processes are regularly adjusted. This causes static instructions to quickly become obsolete, leading to operator confusion or misuse. Here, flexibility is lacking: current systems are rigid and provide no up-to-date guidance. Even when data is available, it is not used in an agile way that enables operator guidance. There is a gap between diagnosis (what happened) and action (what to do about it).

1.2.2 Stakeholders & Challenges

To better understand the context of exception handling in Cyber-Physical Production Systems (CPPS), Table 1.1 and Figure 1.2 outline the main stakeholders involved and summarize the primary challenges each of them faces.

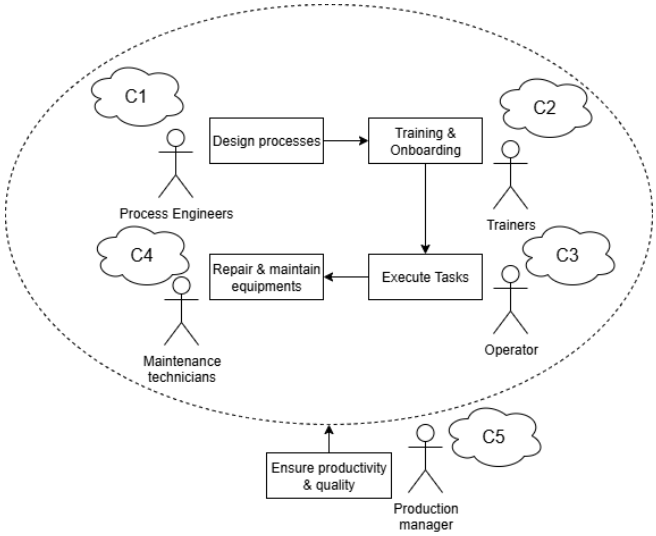


Figure 1.2: Diagram summarizing the key challenges for each stakeholder.

In figure 1.2 we can easily see how the main stakeholders in the production line interact. Process engineers design the processes and support training, trainers help operators learn, and operators execute tasks on the line. Maintenance technicians repair and maintain equipment, while production managers focus on ensuring productivity and quality. Each role has its own challenge (C1–C5), but all are connected through the workflow.

Stakeholder	**Challenge**
Operators	**C1: Struggle to execute and monitor production tasks** Reliance on manuals or experts during exception handling Limited autonomy because of lack of real-time guidance
Process Engineers	**C2: Difficulties to design, optimize, and update production processes** Only static, non-actionable documentation available Difficulty updating guidance with real-time data
Maintenance Technicians	**C3: Troubles when repairing and maintaining production equipment** Unclear or outdated root cause documentation Lack of links between failures and procedures
Production Managers	**C4: Can't ensure productivity and quality** Poor insight into recurrent failures Slow escalation of unresolved issues
Trainers	**C5: Time consuming operator training and on-boarding** Inefficient training procedures Limited adaptability of training materials to system evolution

Table 1.1: Stakeholder Challenge Map in CPPS Exception Handling

Table 1.1 describes how different roles face different problems: operators depend on manuals and experts with little real-time help, engineers struggle with static documents that are hard to update, technicians deal with unclear root cause info and missing repair links, managers lack insight into repeated failures and face slow escalation, and trainers spend too much time on inefficient training that does not adapt to system changes.

1.3 Objective

The general objective is to **provide real-time and contextual guidance** to operators through a user-friendly interface to help them better manage production errors.

This involves translating expert knowledge about error handling into **clear and actionable processes** that can be followed step by step. Another objective is to **allow users to input data** during the entire process about the environment through characteristics. This would greatly facilitate the work of operators when dealing with unexpected situations. The aim is to empower less experienced staff to handle complex exceptions confidently [27].

This overall objective gives rise to several related objectives. By bringing the expert knowledge into a software that can be used live during production, the objective is to **centralize data sources** into a single coherent and understandable knowledge graph. Maintenance, production, and observation data will be merged into a unified graph that can be used by our interface to provide guidance but also by future internal or external tools to enable deeper or more specific analysis. This could include learning from the frequency of defects, using graph structures to detect indirect causal chains not obvious to humans, or doing machine learning on the knowledge to improve future guidance.

A unified source of knowledge enforces **data consistency**, aiming to minimize incoherence through structured ontologies and standard formats. Ensuring **up-to-date data** is also one of our objectives, even if this often leads to inconsistency. This also includes a **version control** across the entire knowledge graph (including guidance materials) to keep a history of the knowledge, allow backward analysis, and recovery procedures in case of problems.

These concrete objectives also imply what can be termed secondary objectives. They are considered secondary in the sense that, while important and desirable, they are either indirectly achieved through other objectives or are long-term goals difficult to measure within the scope of this work.

Process guidance for error handling must **reduce the defect rate**. By systematically capturing expert intervention knowledge, we want to prevent the recurrence of known errors. The guidance will also ensure that best practices (also in terms of root-cause-based corrective actions) are followed consistently across shifts and teams. This digital memory of previously resolved incidents helps to reduce future variability. Even if generated processes are not fully coherent, enabling process adjustments with minimal manual recalibration should help to lower the defect rate.

The effort required for defect analysis must also be reduced. We want to use the knowledge graph to link cause-effect pairs, reduce reliance on heavy manual diagnosis, and free up expert time by scaling their knowledge. This should also improve plant efficiency by accelerating the exception resolution loop, reducing the impact of downtime, shorten the mean time to repair through guided decision support and optimize shift handovers by maintaining shared and centralized knowledge. [13].

We also want to be aligned with the United Nations Sustainable Development Goals (SDGs) [3] by targeting four of their goals. Concerning good health and well being (Goal 3) [4], the aim is to minimize operator stress in environments where uncommon issues arise by reducing ambiguity during problem-solving tasks. We can also decrease downtime frustrations by resolving exceptions faster and more predictably, contribute to a safer workplace by standardizing actions to high-risk anomalies, encourage ergonomic work practices by streamlining processes [21]. Going now to quality education (Goal 4) [5], we are eager to promote lifelong learning opportunities for employees. By encapsulating expert knowledge into reusable formats, operators can learn autonomously by doing, with guidance derived from experienced staff embedded in the process, allowing non-stop learning. We also set goals concerning industry, innovation and infrastructure (Goal 9) [6], by empowering the Industry 4.0 and CPPS environment with a new digital solution that leverages Knowledge Graphs for intelligent decision support, helps eliminate large amounts of manual and paper-based practices, support other digital systems via modular architecture, and familiarizes employees with digital systems by providing a user-friendly interface. Finally, responsible consumption and production (Goal 12) [7], is also an important topic. Our goal is to contribute to more sustainable plants by reducing waste associated with defaults. This includes avoiding repeated mistakes that lead to resource loss, preventing material loss through faster and more accurate response to faults, limiting energy waste by optimizing recovery procedures [11].

1.4 Method

Here we describe the methods applied within the design science framework to analyze capabilities, implement the prototype, and evaluate the APEH solution.

[3] United Nations, *THE 17 GOALS*, 2025. Available at: `https://sdgs.un.org/goals` [Accessed: 2025-07-01].

[4] United Nations, *Ensure healthy lives and promote well-being for all at all ages*, 2025. Available at: `https://sdgs.un.org/goals/goal3#overview` [Accessed: 2025-07-01].

[5] United Nations, *Ensure inclusive and equitable quality education and promote lifelong learning opportunities for all*, 2025. Available at: `https://sdgs.un.org/goals/goal4#overview` [Accessed: 2025-07-01].

[6] United Nations, *Build resilient infrastructure, promote inclusive and sustainable industrialization, and foster innovation*, 2025. Available at: `https://sdgs.un.org/goals/goal9#overview` [Accessed: 2025-07-01].

[7] United Nations, *Ensure sustainable consumption and production patterns*, 2025. Available at: `https://sdgs.un.org/goals/goal12#overview` [Accessed: 2025-07-01].

1.4.1 Design Science

This thesis is structured and conducted according to the guidelines of Engström et al. [24]. The visual abstract, as can be seen in Figure 1.3 adapted to this thesis, captures three main aspects of the design science cycle. (1) the proposed theory in the form of a technological rule; (2) at least one problem-solution pair with design and validation cycles as empirical contribution and (3) guidance for the assessment of the produced knowledge in terms of relevance, rigor, and novelty.

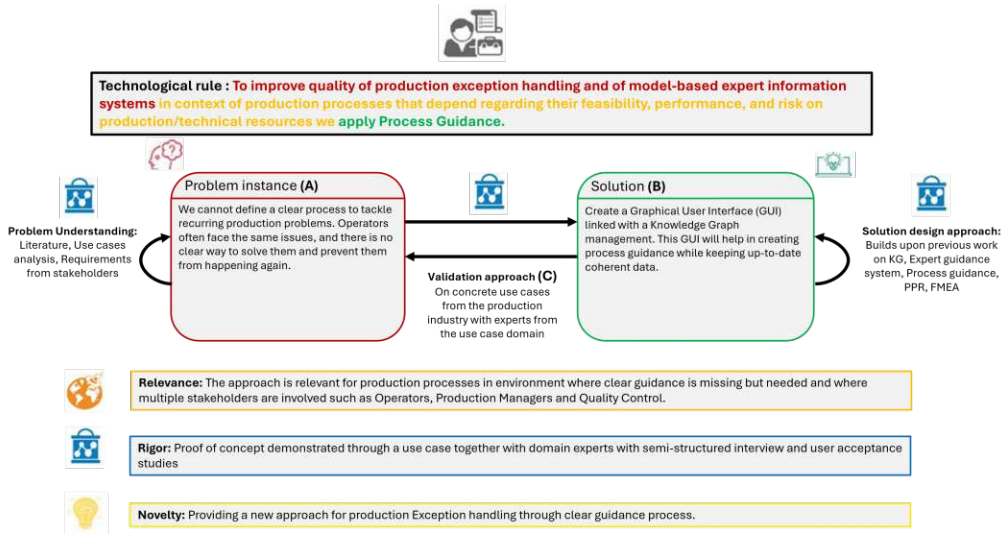


Figure 1.3: Visual Abstract of the proposed research, adapted from Engström et al. [24]

By following this Design Science cycle difficulties were identified for production operators in tackling recurring errors during the production processes, the problem was further explored through literature reviews and requirement analysis. This allowed us to develop our artifact, a graphical user interface that enables the creation of guidance for exception handling and evaluate it through a feasibility study, semi-structured interviews, and user acceptance studies.

1.4.2 Analyze Capabilities

This section corresponds to part **(A)** of our design science cycle.

Systematic Literature Review. To synthesize existing knowledge on production exception handling, FMEA limitations, agile process guidance, and related topics, we first conducted a SLR, which is a structured approach to synthesize data and address certain research problems through transparent and reproducible steps. The guidelines of Carrera-Rivera's [14] & Lame's [40] were followed, which split it into two main phases: planning and conducting. During the planning phase, we define a protocol, formulate focused research questions (often using the PICOC framework), select relevant digital libraries, establish clear inclusion and exclusion criteria, define a quality assessment

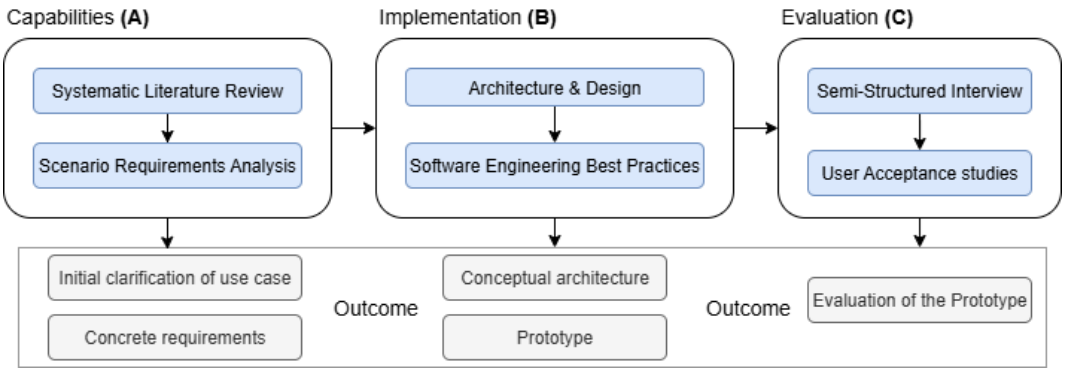


Figure 1.4: Schema summarizing used research methods

checklist, and design a data extraction form. We then operate the conducting phase, which involves building search strings, gathering studies from the selected databases, performing study selection and refinement (removing duplicates, screening based on criteria, and assessing quality), to finally extract data from the included studies.

Scenario Requirements Analysis. Based on what we have learned with SLR, SCRAM was then applied to collect requirements directly from users We will follow guidance from Sutcliffe [71] with a clear path, containing four techniques. First, simple mock-ups of the system were built, shown to users, and receive feedback. Second, we use scenarios, which are like stories showing how the system works for a user's task. This helps users understand how the prototype fits into their real work. Third, design rationale was documented, for example by explaining why certain design choices were made, sometimes using a format like QOC (Questions, Options, Criteria) to show different ideas and why one is chosen. This helps users to discuss the ideas. Finally, we do a whiteboard summary to write down the final requirements.

1.4.3 Implement Prototype

This section corresponds to part **(B)** of our design science cycle.

For the prototype implementation, specific Vocabulary and Color Codes were used and can be found in Appendix

Using the previously defined requirements and mock-ups, we can build, by following the software engineering best practices, the architecture of our software. Here, recent and proven techniques were applied by also applying basic principles of software engineering as well as all the knowledge we acquired during our studies. Requirement Engineering was performed to understand the problem and define the objectives to be addressed [73]. This also helped us to understand the business area. A clear software architecture serves as the blueprint for the system. Here we try to divide the system into smaller, independent parts called "modules" or "components". This makes the software easier to change and fix later, because if one part has a problem, it does not break everything else [73]. n addition,

documentation was prepared for later reuse, and continuous testing was conducted to avoid later modifications. Once the architecture was set up, the solution was implemented in an agile manner by iterating with key stakeholders as a feedback loop. We start by building the overall solution and deep dive with time into specific features allowing more efficient feedback. By using short release cycles, called sprints, important stakeholders were involved in the project, contributing ideas and suggestions to build a higher-quality artifact. [23].

1.4.4 Evaluate the solution

This section corresponds to part **(C)** of our design science cycle.

Semi-Structured Interview. To evaluate the solution in a flexible but structured way, semi-structured interviews were conducted, being very useful to get data in a qualitative way. Questions were planned in advance, but flexibility was maintained to adjust or add questions during the talk based on the flow. This makes it flexible and personal, letting people share their ideas and experiences. Following Nuzhat's [55] guidance, a guide was built, pilot rounds were performed and reliable information from our stakeholders were collected.

User Acceptance Studies. According to Davis [18], two major aspects were assessed: whether people considered the system useful for their job, and whether it was easy to use. We use the concepts of perceived usefulness and perceived ease of use by defining execution scenarios, sharing them with key stakeholders and observing their reactions through rating scales.

1.5 Contributions

The following key contributions were achieved in the course of this thesis:

By conducting our scenario analysis, we collected a list of requirements from domain experts, software engineers, and previous researchers on the topic that such an expert system should fulfill to be valid. This revealed the importance of PPR assets, their characteristics, live monitoring of process states, and the overall flexibility required when dealing with exceptions. This ensures that the proposed solution addresses real-world needs and constraints, thereby increasing its practical relevance.

We designed and implemented an APEH solution with an **information system that generates process guidance for exception handling while enabling agile flexibility to modify processes on the fly to better treat exceptions**. By adapting to an existing solution, we demonstrate technical feasibility, modularity, and integration into existing production IT systems.

In addition, we created a **domain-specific Knowledge Graph** enriched with external and historical data sources that enables semantic reasoning and dynamic process adap-

tation, informed by a richer context of exceptions. This will also contribute to further research that can use these graph data to perform further detailed analysis.

Beyond demonstrating the technical feasibility of our solution, we proposed a **novel semi-automatic approach to generate agile, context-aware process guidance by enabling the system to adapt existing guidance rules with new knowledge and data over time**.

We also ensured that our solution is **compatible with an existing standard framework** (FMEA) and and demonstrated how it can be leveraged as a basis for our process guidance & exception handling to facilitate root cause analysis and promote proactive resolution strategies. This opens a new path of research that combines the best of both worlds.

Finally, we evaluated and validated our solution with various profiles from the domain. We demonstrated that our solution is useful in practice and easily usable, even for stakeholders unfamiliar with the system. We also identified limitations in particular with high disruption and discovered that adaptations had to be made to be able to use the prototype in a real-world environment.

Based on these contributions, this work supports the following communities:

- **Academic Researchers** in the domain of CPPS can benefit from this work because it presents a real-world application of semantic technologies, agile software engineering and human-centered design in the manufacturing domain, providing a concrete reference for future studies on smart factories and operator assistance systems
- **Manufacturing Operators** can benefit from this work because it provides an intuitive step-by-step, guidance system that reduces the reliance on outdated paper manuals and expert intervention, enabling faster, safer, and more autonomous exception handling in real-time.
- **Process Engineers** can benefit from this work because it offers a way to transform static FMEA knowledge into actionable workflows, supporting systematic fault reduction, improved root-cause analysis, and standardized responses.
- **Plant Managers** can benefit from this work because it offers a ready-to-deploy process guidance prototype that can increase production line resilience, lower rework costs, and accelerate exception resolution.

1.6 Structure of the Work

To explain the research process and the results in a clear way, this thesis is divided into several chapters.

Chapter 2 gives an introduction to the main concepts related to this work. It also presents some previous research and work in the same area. This helps us to understand the background and how this work fits into the existing literature and work.

After that, Chapter 3 explains the research questions. These questions guide the whole work and show what we want to find out about process guidance, how to build the system, and how to test it in a real environment.

Chapter 4 answers the first research question. It focuses on what a semi-automatic process guidance system should be able to do. We look at which features are needed and why they are important for helping workers in the industry.

Then in Chapter 5, the prototype system is described. We explain why we made it this way, how we built it, and which functions it has. This chapter is related to the second research question.

In Chapter 6, we test the prototype in a CPPS environment. We describe the setup, how the evaluation was performed, and what the results are. This chapter helps to answer the third research question.

Chapter 7 goes over all the findings. It discusses how well the research questions were answered and also points out some limits of the current solution.

Finally, Chapter 8 gives a short summary of the most important results and suggests some ideas for future research.

In appendix, we can find additional materials, especially the Vocabulary and Color Codes helping to better understand this work.

CHAPTER 2

Related Work

This section provides an overview of the main concepts and approaches relevant to this thesis. We first look at Cyber-Physical Production Systems (CPPS) and how they support smart manufacturing. Next, we examine the role of asset network PPR models in organizing production resources and processes. We also review how FMEA is used for quality assurance, and finally, we cover some work on process guidance systems. These topics form the foundation for the work presented in the next chapters.

2.1 Cyber Physical Production System (CPPS)

In this section, CPPS are explained, with implementation details and associated challenges.

2.1.1 What are CPPSs?

CPPS are very important for Industry 4.0 and smart factories [49]. They combine computer systems with physical processes and human and social parts [3]. They can be understood as smart systems that communicate with each other and to the real world, using the Internet to share and process data [49]. CPPS have parts that can work by themselves or together. These parts connect at all levels of production, from the processes through machines up to the production and logistics networks [49]. They can be seen as a projection of Cyber-Physical Systems (CPS) into the factory domain. [6].

These intelligent and connected systems are used across multiple sectors and applications. In manufacturing and production, CPPS are used in "smart factories" for tasks such as robotics on assembly lines and machining operations (e.g., milling, turning, honing in the automotive sector) [32], enabling autonomous processes and the update of older industrial equipment. Beyond traditional factories, they are used in the energy sector and smart grids for real-time power management and distribution, as demonstrated

by their application in hydroelectric power plants. CPPS also transform healthcare, particularly through smart medical devices and Ambient Assisted Living (AAL) for patient monitoring and support, and are vital in logistics for optimizing supply chain processes such as automated stock determination and order placement [32].

These systems are widely adopted to foster the development of intelligent automation, real-time monitoring, and comprehensive optimization of industrial processes. These systems enable environments to become resilient and self-adaptive, supporting flexible, reconfigurable, and transformable manufacturing paradigms [79].

Humans play a central and evolving role within these systems, rather than being replaced by technology [79]. CPPS are fundamentally designed as human-centered systems, fostering deep interaction and collaboration to achieve common goals, moving beyond user-centered designs that regard humans merely as technology users [3]. This transition means that job roles shift from low-level operations to higher-level decision-making, requiring new skills to customize machine performance under various conditions. New methods for human-machine communication are continuously being investigated [49]. Frameworks like the PSP Ontology aim to identify optimal collaboration between humans and CPPS in problem-solving by analyzing problem structures, human competence, CPPS autonomy, and solution maturity [3]. Humans demonstrate the ability to handle unstructured and semi-structured knowledge with varying processing times depending on their competence and the volume of knowledge. Visual aids, such as agent-based 3D visualization tools on platforms like INVERSIV, provide crucial support to human users by visualizing repair actions for operators and workers based on diagnosed faults [81]. Moreover, CPPS processes are envisioned to effectively incorporate humans to facilitate the sharing of opinions, experiences, and requirements, enabling discussions about new product designs.

2.1.2 CPPS design and operation

One key framework for developing and deploying CPPS in manufacturing is the 5C architecture, which offers a step-by-step guideline [42]. This architecture outlines how to construct a CPPS from initial data acquisition to value creation, through a sequential workflow. As shown in Figure 2.1, it consists of five layers

The Smart Connection Level (I), focuses on acquiring accurate and reliable data from machines and their components, emphasizing data acquisition and transfer to central servers using appropriate sensors and protocols like MTConnect[1]. The Data-to-Information Conversion Level (II) then employs smart analytics for various purposes, including monitoring component machine health, correlating multi-dimensional data, and predicting degradation and performance. The Cyber Level (III) acts as a central information hub, where vast amounts of information from connected machines form a network. Here,

[1] MTConnect Institute, *MTConnect standardizes factory device data*, 2025. Available at: `https://www.mtconnect.org/` [Accessed: 2025-07-09].

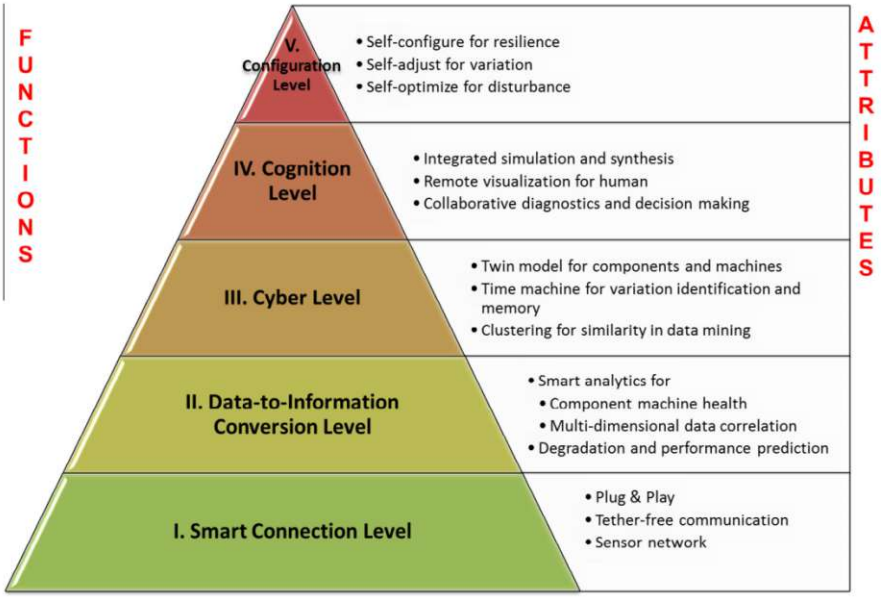


Figure 2.1: 5C architecture for implementation of Cyber-Physical System from Lee et. al

specific analytics are applied to extract insights, facilitating self-comparison among machines and predicting future machinery behavior based on historical data. This level can be conceptually similar to social networks, enabling machines to register and exchange information. The Cognition Level (IV) supports integrated simulation and synthesis, remote visualization for human users, collaborative diagnostics, and decision-making, aiming to transfer knowledge to users by presenting the monitored system. Finally, the Configuration Level (V) enables the system to self-configure for resilience, self-adjust for variation, and self-optimize for disturbances, by applying decisions made at the cognition level back into the physical world [42].

When designing CPPS, we include principles such as interoperability, where CPPS and humans are connected over the Internet of Things (IoT) and Internet of Services, communicating through open networks and semantic descriptions [43]. Virtualization links sensor data to simulation models, creating virtual copies of the physical world for process monitoring. Decentralization allows embedded computers within CPPS to make autonomous decisions, shifting from traditional hierarchical architectures to more distributed control [43]. This interconnected design supports information transparency, aggregating raw data into information of higher-value in contextual cyber space, and decentralized decisions, empowering CPPS and employees with personalized information for autonomous action [39].

This interoperability enables the gathering of large amounts of data, prompting researchers to explore solutions for its effective use. For example, a knowledge-based CPPS design approach can proactively provide the needed design knowledge to minimize or avoid future disruptions [25]. This knowledge needs to be provided at the right time and with

the right context. A prototype tool that uses this idea, called a digital factory tool, has shown that it helps people see the effects of their choices, which is very useful in practice [25]. The digital factory concept helps integrate the planning, evaluation, and ongoing improvement of the real factory using virtual environments. Computational digital factory tools are a good way to handle the complexity of CPPS design. Certain support systems combine automatic reasoning, domain-dependent computations, and user consultation for complex reconfiguration processes. For example, a knowledge-based approach for the reconfiguration of automation systems helps users in re-planning tasks when new requirements come up [25]. However, some existing platform-based design methods for Cyber-Physical Systems (CPS) help to check requirements early, but do not give proactive guidance to the user.

CPPS are human-centered systems. This leads to a shift from just human-machine cooperation to active collaboration [3]. This means humans and CPPS become "problem-solvers" together, sharing knowledge and learning from each other. New ways for people and machines to communicate are needed to establish CPPS. To figure out the best way for humans and CPPS to solve problems together, researchers have introduced the Problem-Solving Profile (PSP) Ontology [3]. This semantic framework helps to identify the optimal collaboration between human and CPPS in problem-solving.

When it comes to maintenance of these systems, there is a new approach called Prescriptive Maintenance Model (PriMa) [2]. This moves from just describing what happened to actually giving recommendations for optimizing future maintenance plans. PriMa has four layers: data management, predictive data analytic toolbox, recommender and decision support dashboard, and an overarching layer for semantic-based learning and reasoning [2]. This is part of a larger idea called Knowledge-Based Maintenance (KBM) [60], which uses advanced statistics, machine learning algorithms, and semantic modeling and representations to continuously support value generation and facilitate the development of collective knowledge in smart factories [2].

Another important research topic is Digital Twin. These are virtual copies of real production systems that help with optimization and real-time production control for CPPS [72]. They can help protect knowledge by separating volatile data and master data, forming a system of inherent data security. Automated data acquisition and selection are needed for near real-time evaluation and analysis of highly complex production systems [72]. This makes it particularly useful for enhancing decision-making, reducing downtime, and enabling predictive maintenance in smart manufacturing environments.

2.1.3 CPPS concerns

Several research challenges and gaps exist in the development and implementation of CPPS. The increasing complexity of manufacturing control systems demands new tools for monitoring, failure detection, and analysis [81]. Technological challenges include integrating heterogeneous cyber-physical elements, addressing real-time data collection and physical interfaces, and effectively designing human interfaces, as-well as effectively

designing human interfaces and integrating human factors [2]. Another significant challenge lies in extracting knowledge from heterogeneous, multi-modal data with varying quality and credibility, which requires advanced data analytics and Artificial Intelligence techniques. In the field of cybersecurity, analyzing encrypted data, securing analytics platforms, and integrating real-time data streams for autonomous maintenance workflows, also remain critical obstacles. Data quality also remains often insufficient for advanced simulation [72]. Achieving interoperability among diverse CPPS components and overcoming the 'standardization deadlock that makes information exchange processes more difficult also present ongoing issues [6]. Concerning fault diagnosis, current approaches struggle with complex systems, and model-based approaches require manual modeling of causality, creating bottlenecks [53]. Models that capture hybrid and timing aspects and focus on component-level anomaly detection over root cause analysis are needed [53].

In this work, we focus on the integration of heterogeneous issues with real-time data collection and human factors and the homogenization of data sources to tackle challenges 1 and 2 from our Table 1.1.

2.2 Product-Process-Ressource (PPR) Asset Network

In this section, we first describe PPR assets, show some of their applications in industry, and share identified research gaps.

2.2.1 Product-Process-Ressource (PPR)

PPR PPR is a structured approach to conceptualizing how products are manufactured in a factory. It helps different experts, who work on designing production systems, to reuse elements developed in previous projects [46].

A PPR asset can be many things, for example, a car body (as product) or "joining parts" (as process) [10]. It also includes resources, which can be physical things (machines or sensors) but also non-physical (recipes or computer programs) [46]. Each of these assets has properties—a collection of details used by engineers from different disciplines, such as mechanical or electrical, to describe the assets. For example, a "Screwing" process asset can have properties about its quality [10].

In more detail, the PPR model has three main parts [65]:

- Product: This part holds all the information about the product, its features, and if it has different versions. A product can be made of many parts that are connected in different ways.
- Process: This part has knowledge about the steps needed to make the product. These steps use resources to get the work done. Some processes also need special skills from the resources or people.

- Resource: Describing needed assets during processes. This can be physical tools or materials, and non-physical things, like the knowledge that helps the production process.

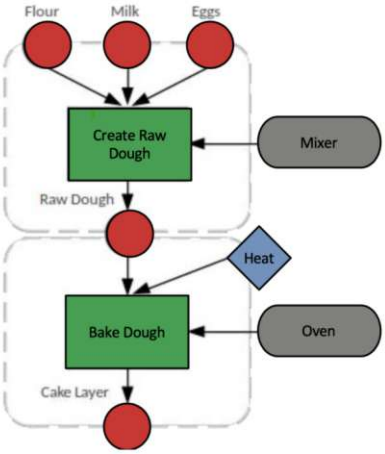


Figure 2.2: Illustration of the Product-Process-Ressource model for a simple cake baking workflow

As illustrated in Figure 2.2, the three products (flour, milk, and eggs, shown in red) feed into the "Create Raw Dough" process in green using the mixer resource in Grey.

These PPR assets can be linked to form a PPR Asset Network (PAN) [77]. This network shows how different assets are connected and depend on each other. This serves as a coordination artifact.

PAN is used because it gives a complete and integrated picture of the production system. This helps engineers and experts better coordinate their work in different disciplines and manage complexity [7]. For example, it is used for tasks such as integrated risk analysis, identifying root causes of issues, managing changes, or systematically specifying tests [37]. Although setting up a detailed PAN requires effort, particularly at the beginning, it significantly improves efficiency for recurring tasks by providing a clear and connected view of production reality [78].

2.2.2 PPR Industry Applications

One application of PPR models is to help engineers work more efficiently. When designing production systems, many different teams must work together. They often have different ways of seeing things and use different tools. PPR helps to bring these different viewpoints together [77].

It helps to organize knowledge and transfer it from early planning to more detailed engineering and even during factory operation. This is important because without good coordination, changes in one part of the design can cause problems, leading to rework

and delays. PPR models try to make these connections clear, so everyone knows how changes affect other parts [10]. For example, in welding robots, a PPR model can store all the needed information in one place, so it is faster to program the robots and needs less human work, which can ave up to 80% of the time [65]. PPR can also help to make process planning faster and better by linking product, process, and resource information automatically [76].

PPR is used in many industries, especially in complex ones such as automotive manufacturing. For example, it is used for adapting robot cells that join car parts [46], for automatic welding-robot programming or in assembly lines, when making automotive engines [65].

In practice, PPR models can be put into special databases, like a graph database called Neo4J [46][2]. This makes it easy to look at and ask questions about the PPR information. For example, you can see how an electric screwdriver resource is connected to a "Fasten Screw" process and the "Car Body with Engine" product it works on [10]. Researchers are also working on using PPR to manage risks, trace requirements, and support configuration management [46].

PPR models are also integrated into existing standards:

- The PPR Asset Network (PAN) is a kind of metamodel that the MRC artifact extends. It's based on the Industry 4.0 Asset Network design [46].
- STEP AP214 standard was used to represent PPR information for machining lines in the automotive industry. This was valuable because it enabled the sharing of information in a standard way between different companies and computer systems [75].
- PLCS (AP239), is more general. It can show resources in different life cycle stages, like how they were designed, planned, or how they are now in the factory [75].
- The VDI/VDE 3682 series of standards provides a formalized framework for process descriptions, designed to model technical processes across the entire technical systems lifecycle. It defines a simple symbolism for objects, to form a network [74].

2.2.3 Concerns with PPR models

Researchers are actively working on enhancing PPR models. For instance, the Multidisciplinary Reuse Coordination (MRC) artifact is being developed to better manage multidisciplinary reuse tasks [46]. Another example is the FMEA-linked-to-PPR Asset Issue Analysis (FPI) Model, which helps guide quality issue analyses by linking failure modes to production assets [38]. Work on a Digital Twin concept for Process Planning

[2]Neo4j Inc., *Neo4j – The Leader in Graph Databases*, 2025. Available at: `https://neo4j.com/` [Accessed: 2025-07-14].

(DTPP) that links product, resources, and processes to automate planning is also in progress [76].

However, PPR needs to take into consideration the following aspects:

- Scattered Knowledge: Even with PPR, information often remains distributed across many different documents and systems, making it hard to find all the details you need [46]. Traditional task management systems have trouble with this because they don't always understand the exact meaning of engineering objects.
- Scalability: For large factories, such as automotive plants, with hundreds of robot cells, the PPR network can become huge and very complex. It is hard to manage all this information and track everything efficiently [10].
- Security Concerns: PPR models bring together a lot of important company knowledge, raising concerns about security and the risk of malicious attacks aimed at stealing data [46].
- Implicit Knowledge: Much expert knowledge remains undocumented. PPR models aim to make this knowledge clearer, but it is an ongoing effort [10].
- Standardization Issues: Certain PPR standards remain ambiguous, leading to different ways of representing the same information. This makes it difficult to have consistent data across different projects or companies [75].

As PPR asset network explicitly shows dependencies and states of assets, this data can be used to understand the causes of exceptions, the affected parts, and the corrective steps to be taken. This becomes a strong base to create the semi-automatic guidance for unexpected situations. Without this organized PPR information, it would be much harder to quickly obtain the right details to handle problems in an agile production environment.

Here, we focus on Scattered Knowledge, Scalability, and Standardization Issues addressing challenges 4 and 5 of our Table 1.1.

2.3 Quality assurance

In the era of Industry 4.0, CPPS help factories become more flexible and capable of producing high-quality products. Quality managers in these settings aim to identify potential quality problems effectively and efficiently. For example, in car manufacturing, systems for fastening parts are particularly critical. If a screw joint is not tight enough, it can make the car unsafe [9].

By using various sensors, CPPS can monitor product quality in real time. A Cyber-Physical Quality System (CPQS), for example, can predict and check quality with 95% accuracy in real-time using machine learning. This system analyzes sensor data such

as speed, temperature, vibrations, and cutting forces [17]. This helps prevent quality problems during production, enables faster defect detection, and improves material utilization.

Neal [52] outlines a case study in which a CPPS was used in an automotive factory to monitor the washing of special containers called Returnable Transit Items (RTIs). These RTIs must be clean to prevent contamination of engine parts. The system used small RFID tags to track the RTIs and their plastic separators. This helped to make sure every separator was washed after a certain number of uses. The aim is to make production more efficient and reduce costs by preventing failures.

2.3.1 Failure Mode and Effects Analysis (FMEA)

To ensure quality during production cycles, plant managers use frameworks that set best practices for analysis. For the scope of this work, we focus on one of them: the Failure Mode and Effects Analysis (FMEA). This framework is well known and widely used within the automotive industry as it is also a requirement for most of the quality standards in the domain (ex: DIN 60812 [9], DIN 13379-1 [20]).

FMEA is an engineering method that helps to find, identify, and offer solutions to system failures, problems, or errors [58]. FMEA supports decisions aimed at preventing single system failures. It is a proactive, team-based way to reduce risks and improve safety [62]. FMEA has been used for a long time, since the 1960s. It has been applied in many industries, including nuclear, military, aviation, food, and automotive. Now, it is also used in healthcare and other service industries [62]. The main goal of FMEA is to prevent system failures from the outset, thereby saving money and time. FMEA also records what is known about failure risks, which helps reduce their impact [67]. It can be used from the earliest design stages, through development and testing, and throughout the product's entire life [9].

FMEA defines key steps for a successful analysis:

- **Identify potential failures and effects**. This means listing all the ways a system or component could malfunction and what the consequences would be [58].
- **Determine severity**. This is about how serious the outcome would be if a failure happened. Severity is often scored, where a low score means that only a minor part is affected, and a high score means that it could cause injury or death [67].
- **Gauge likelihood of occurrence**. This step looks at how often a failure is likely to occur. Companies often look at similar processes or products to estimate this [67].
- **Failure detection**. This assesses how easily a failure can be detected before it causes harm to the system or end user. A higher score here means it is harder to detect the problem [20].

After these steps, a Risk Priority Number (RPN) is calculated for each potential failure mode. The RPN is calculated by multiplying the scores, occurrence, and detection [58]. This RPN helps to prioritize identified problems. It helps teams focus their efforts and resources on the most critical issues first [62]. For example, a table might show components ranked by RPN, with the highest RPN indicating where technicians should start looking for a problem.

FMEA is a strong method for risk analysis. It performs well for complex systems and effectively identifies technical failure modes. It is proactive in reducing risk by early identifying potential failures [62]. This helps prevent problems and improves quality before issues reach customers. It uses a systematic approach, offering a structured method to analyze failures and their effects [58]. It is versatile and used across various industries, including nuclear, military, automotive, and healthcare [62]. It applies throughout the entire product life-cycle [67]. The Risk Priority Number (RPN) helps in prioritizing failures, guiding teams to focus resources where the impact is greatest.

Despite its strengths, FMEA has limitations, especially with modern Cyber-Physical Systems. Its results are heavily dependent on the team's experience and knowledge. This can lead to biased or inconsistent results [20]. Conducting FMEA is time-consuming and labor-intensive, especially for large and complex systems [20]. It tends to focus on single failure modes and may miss combined or interacting faults in mechatronic systems [9]. The standard RPN calculation treats severity, occurrence, and detection equally, which may not always be suitable [58]. It does not account for economic impacts such as social costs, infrastructure losses, environmental damage, or delays. When FMEA is applied late in development, fixing design flaws becomes costly. Updating FMEA after design or process changes is difficult and often involves manual effort [63].

2.3.2 Industry applications

New research aims to improve FMEA, especially for CPPS. A significant development is the move towards data-driven FMEA. This involves using Artificial Intelligence (AI) and machine learning with data from sensors and past operations [20]. The goal is to use AI and machine learning on historical and real-time data to make RPN calculations more objective. For example, deep learning models have been used to predict failure occurrence in the aviation industry with high accuracy (around 95%). This makes the risk assessment less dependent on human experience. Data-driven FMEA also helps in maintenance planning by combining insights from past events with employee experience. This means that every employee using the system gets the same, objective result [20].

A new version of FMEA, called FMEA-MSR (Monitoring and System Response), is being developed. This approach is designed to specifically address the risk analysis of complex mechatronic systems, which traditional FMEA did not cover in detail [28].

Researchers are also working to link FMEA with other system models, such as PPR models, in a multi-view way. This helps to understand how changes in one part of the system can affect others across different engineering disciplines. The CPPS Risk Assessment

(CPPS-RA) approach, for instance, links FMEA elements to a CPPS Engineering Network (CEN). This allows for validating cause-effect hypotheses by connecting them to actual CPPS data [9].

Creating automated FMEA/FMSA systems that can continuously update risk and monitoring priority numbers is another current point of interest. This could help manufacturing systems become self-regulating for maintenance based on real-time data [20].

Some studies propose new RPN calculation methods that include economic impacts like social, infrastructure, environmental, and delay costs, to provide a more holistic view of risk [58]. Current FMEA tools, like APIS IQ [3], typically focus on documenting FMEA concepts in a textual format. While useful for recording information, these traditional tools and methods are often document-based [63]. Those tools struggles dealing with current challenges already mentioned before in this work, like re-evaluation when processes changes or subjectivity related to expert knowledge. Further than that, they do not easily link FMEA concepts to real-world physical system data, sensor data, or other engineering models. This makes it difficult to validate hypotheses with actual operational data [63].

2.4 Process Guidance

Process guidance is a system that helps people perform tasks correctly by giving them information or suggestions [50]. Multiple terms can be used to describe such systems, including 'Task Guidance', 'Integrated Operator Decision Aid', 'Operator Assistance Systems', and many more. These systems are integrated into CPPS and more generally into Industry 4.0 to collect, process, and share information to help workers in industrial tasks [64].

To enable this guidance, a process guide is needed, which is a document that helps people follow a process. It gives instructions on how to do the work [35]. Usually, we think of a paper checklist, but the objective of process guidance is to extend this and enable much more than that. [56].

Process guidance systems are made to support procedural tasks. These tasks are common in almost every industry for example : inspection, maintenance or assembly [56]. In the past, people focused on making the procedure itself or automating the worker out of the task. This idea of helping users with a guidance system is a newer way of thinking where we want to give more importance to human and place it in the center of the production processes. Humans provide significant added value, which can be fostered through appropriate guidance.

These systems include different types of information. They always include process definitions showing what activities need to be done, what products or "artifacts" are

[3] APIS Informationstechnologien, *APIS IQ-Software Version 8.0*, 2025. Available at: `https://www.apis.de/en/software/products/apis-iq-software-version-8-0` [Accessed: 2025-07-01].

involved, and who is responsible for the work, like roles or agents. They also explain how all these parts are connected [35]. They can also give more details and tell you the main goal of a task, why each step is important, or what conditions are needed for the task [56]. Sometimes, they provide very specific instructions, while in other cases they offer more general advice. The guide can even change based on the situation, like if something unexpected happens [56]. These systems can also understand how steps depend on each other meaning workers can be flexible with the order of tasks, but the system still makes sure critical steps are finished first [56].

For designing these systems, experts talk about three types of knowledge they need to support [50]:

- Process orientation knowledge, helps users know where they are in the whole process. It helps them find their current position.
- Process overview knowledge, gives a simple summary of all the steps and their sequence.
- Procedural process knowledge, tells users exactly how to do a specific step.

Guidance is useful for many reasons. People can be new to the process, forget things, or might need to change their actions based on what is happening now. Guidance helps them complete tasks successfully and helps record terminated tasks with completion state. Procedures are very important for complex tasks, they give a clear, efficient, and tested way to work [56].

In industry, process guidance helps with many challenges. This includes meeting customer needs, dealing with complex products, keeping machines and workers productive, and training people [48]. It helps reduce errors and makes tasks less mentally demanding. It also helps workers gain more knowledge about their tasks and perform better [16, 50]. This is especially true for humans who are increasingly handling more difficult and unusual situations as automation grows. Process guidance helps people feel like they are "co-creators" in their work, not just passively taking orders from a machine [5]. It can also improve data quality, for example, by automatically capturing maintenance activities instead of relying on manual input, which can be messy and slow [5].

2.4.1 Process Guidance Applications

Guidance systems also help in specific situations. For example, they can help during pre-flight inspections of airplanes [57], in nuclear power plants [56] or in manufacturing by supporting repair and maintenance tasks [16]. They can also support optimization and real-time production control [16].

In IT Service Management (ITSM), a system called ITSM ProcessGuide helps with managing service tickets [50]. In software engineering, electronic guides help developers

with complex tasks like fixing bugs or handling customer support [50]. In healthcare, it helps surgeons during operations [16].

If we take a closer look at an example from the car industry, when making car body parts, a very sensitive process called deep drawing is used. In this use-case, small changes can cause defects such as tearing or wrinkles. When making a "spare wheel well" for a car, if the material flow is not right or the tools get too hot, the part can be bad [36].

A modern process guidance system would have a process monitoring part and a decision-making assistance part [36]. Sensors on the machine can measure things like the material flow (draw-in) and tool temperature in real-time. If the monitoring part sees something is off, like the material not flowing enough, it tells the operator [36]. Then, the decision-making assistance comes in. It uses complex models, sometimes even artificial neural networks (ANN), trained with data from many simulations, to understand what went wrong. It can then suggest to the operator how to adjust settings, for example, by changing the height of little pieces called adjustable spacers. This helps to control the material flow and stop defects. This kind of system helps stabilize the production process and ensure the parts are of good quality [36].

Research Prototypes

- ARGUS, a system that helps developers build and fix intelligent AR assistants. It shows information from many sensors (video, audio, etc.) and also the output from machine learning models in real-time or from past recordings. [16].

- TIM Personal Assistant, another AR framework that brings together how machines see and hear the world, how they pay attention, and how they understand the user, all for real-time tasks using headsets like HoloLens 2. It uses AI to figure out what actions are happening and what step of a task the user is on [16].

- Knowledge-Based Systems, for example, a system called EULE visualizes a process and gives information on how to perform tasks that users might not know well [50].

- Context-Sensitive Assistance, CONTASK tracks what a user is doing in a system and tries to guess their next steps, giving them helpful information proactively. AssistantPro also gives guidance based on the current situation [50].

- Operator Assistance Systems (OAS) / Cyber-Physical Operator Assistance Systems (COAS), these are broad terms for systems that help industrial workers [64, 47]. They cover five main areas: task guidance (the most common, over 50% of research) [64], communication (like video calls for remote help), knowledge management (capturing and sharing what workers learn) [48], monitoring, and decision-making [64].

Commercial Solutions

In recent years, several commercial platforms have emerged to address the need for process guidance in industrial settings. These systems are designed to improve efficiency, accuracy, and traceability in manual or semi-automated tasks, aligning with Industry 4.0 objectives. Among the most adopted solutions are Operations1, Arkite and VKS. Each platform offers specific strengths depending on use case, integration requirements, and industry context.

- Operations1 is a leading digital work instruction platform that aims to replace paper-based Standard Operating Procedures (SOP) with dynamic, media-rich process descriptions. The platform supports real-time feedback, analytics, and integration with ERP and MES systems. Its user-friendly interface and no-code editor make it suitable for both training and operational environments, especially in discrete manufacturing industries such as automotive and electronics [4].

- Arkite provides an Augmented Reality (AR)-based Operator Guidance Platform that transforms manual workstations into interactive environments. Using projection and sensors, it guides workers in real-time, helping to avoid mistakes and enforce procedural compliance. Arkite is particularly useful in complex assembly processes and supports human-machine collaboration by adapting instructions to real-time process data [5].

- VKS (Visual Knowledge Share) provides a visual standardization platform that combines digital work instructions, live process tracking, and performance analytics. Its drag-and-drop interface allows quick deployment and modification of SOP, while also supporting barcode scanning, user checklists, and IoT integration. VKS is widely used in high-mix, low-volume environments and helps organizations reinforce quality and consistency [6].

2.4.2 Concerns with Process Guidance

The strengths of process guidance systems are many. They are great at reducing errors and making tasks less mentally tiring because they provide the right instructions when and where you need them [16]. They help you to execute tasks more accurately and quickly [50]. They also help you learn more about the processes and better understand them [50]. For complicated jobs, they give a clear, and proven way of working [56]. Modern systems are easy to update and can be accessed by many people over the internet [35]. They put the human worker at the center of the production system [47]. They also improve the quality of the collected data during tasks, which is very helpful to know how well things are going [5].

[4] https://operations1.com/en/software/digital-work-instruction-operations1
[5] https://arkite.com/operator-guidance/
[6] https://vksapp.com/work-instruction-software

However, there are also some weaknesses. Users can sometimes trust the computer too much, if the system does not have all the right information, this can be dangerous [57]. Also, many systems only tell you what to do, not why it needs to be done or how it fits into the bigger picture. This means that users might not use their own higher-level thinking skills [57]. Another issue is user acceptance. Workers might not want to use systems that tell them how to perform tasks they already know well [47]. There can also be a fear of losing jobs due to automation, so systems need to show they are there to help, not replace [48, 5]. Data collected manually, especially for maintenance, can be wrong or inconsistent, making the guidance less reliable [5]. Bechinie [5] also mentioned that not enough research has looked at how to ensure these intelligent systems provide a good user experience (UX) in the long run.

Our main concern here is to create reliable guidance that supports workers through their daily issues, addressing challenges 1 and 4 of our Table 1.1.

2.5 Previous Work

Our work will build upon work from Gruber [29] and Moser [51], two bachelor thesis made in collaboration with the CDP

Looking at previous work from colleagues, we can analyze the existing ATG solution to summarize its key features and deduct the missing ones. The as-is solution allows to import "orders" (Figure 2.3) from a google spreadsheet (Figure 2.4) into the ATG system.

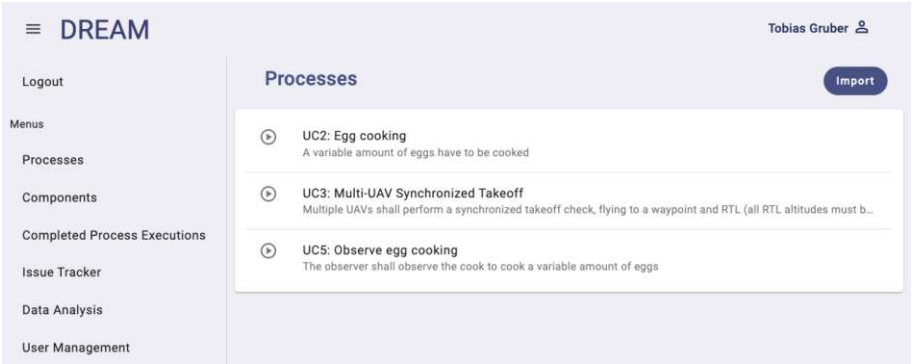
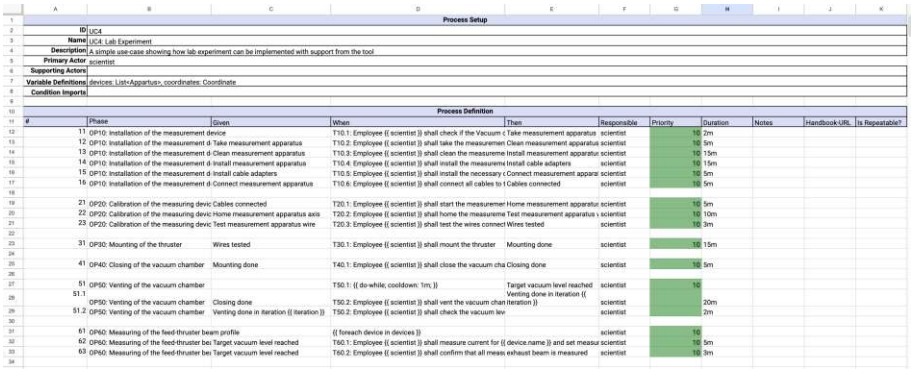


Figure 2.3: User interface for importing

In Figure 2.3 we can see the interface where we import processes from the google sheet and can then instantiate them in the ATG system.



Process Setup

ID	UC4
Name	UC4: Lab Experiment
Description	A simple use-case showing how lab experiment can be implemented with support from the tool
Primary Actor	scientist
Supporting Actors	
Variable Definitions	devices: List<Apparatus>, coordinates: Coordinate
Condition Imports	

Process Definition

#	Phase	Given	When	Then	Responsible	Priority	Duration	Notes	Handbook-URL	Is Repeatable?
11	OP10: Installation of the measurement device		T10.1: Employee {{ scientist }} shall check if the Vacuum c	Take measurement apparatus	scientist	10	2m			
12	OP10: Installation of the measurement d	Take measurement apparatus	T10.2: Employee {{ scientist }} shall take the measuremen	Clean measurement apparatus	scientist	10	5m			
13	OP10: Installation of the measurement d	Clean measurement apparatus	T10.3: Employee {{ scientist }} shall clean the measureme	Install measurement apparatus	scientist	10	15m			
14	OP10: Installation of the measurement d	Install measurement apparatus	T10.4: Employee {{ scientist }} shall install the measureme	Install cable adapters	scientist	10	15m			
15	OP10: Installation of the measurement d	Install cable adapters	T10.5: Employee {{ scientist }} shall install the necessary c	Connect measurement appara	scientist	10	5m			
16	OP10: Installation of the measurement d	Connect measurement apparatus	T10.6: Employee {{ scientist }} shall connect all cables to t	Cables connected	scientist	10	5m			
21	OP20: Calibration of the measuring devic	Cables connected	T20.1: Employee {{ scientist }} shall start the measuremer	Home measurement apparatu	scientist	10	5m			
22	OP20: Calibration of the measuring devic	Home measurement apparatus axis	T20.2: Employee {{ scientist }} shall home the measureme	Test measurement apparatus	scientist	10	10m			
23	OP20: Calibration of the measuring devic	Test measurement apparatus wire	T20.3: Employee {{ scientist }} shall test the wires connect	Wires tested	scientist	10	3m			
31	OP30: Mounting of the thruster	Wires tested	T30.1: Employee {{ scientist }} shall mount the thruster	Mounting done	scientist	10	15m			
41	OP40: Closing of the vacuum chamber	Mounting done	T40.1: Employee {{ scientist }} shall close the vacuum cha	Closing done	scientist	10	5m			
51	OP50: Venting of the vacuum chamber		T50.1: {{ do-while, cooldown: 1m, }}	Target vacuum level reached	scientist	10				
51.1	OP50: Venting of the vacuum chamber	Closing done	T50.2: Employee {{ scientist }} shall vent the vacuum cham	Venting done in iteration {{ iteration }}	scientist		20m			
51.2	OP50: Venting of the vacuum chamber	Venting done in iteration {{ iteration }}	T50.2: Employee {{ scientist }} shall check the vacuum lev		scientist		2m			
61	OP60: Measuring of the feed-thruster beam profile		{{ foreach device in devices }}		scientist	10				
62	OP60: Measuring of the feed-thruster be	Target vacuum level reached	T60.1: Employee {{ scientist }} shall measure current for {{ device.name }} and set measur		scientist	10	5m			
63	OP60: Measuring of the feed-thruster be	Target vacuum level reached	T60.2: Employee {{ scientist }} shall confirm that all meas	exhaust beam is measured	scientist	10	3m			

Figure 2.4: Previous spreadsheet definition for an order

On this google spreadsheet we define processes with the Gherkin language by defining their key properties and relationships. This data is then collected and imported in the system.

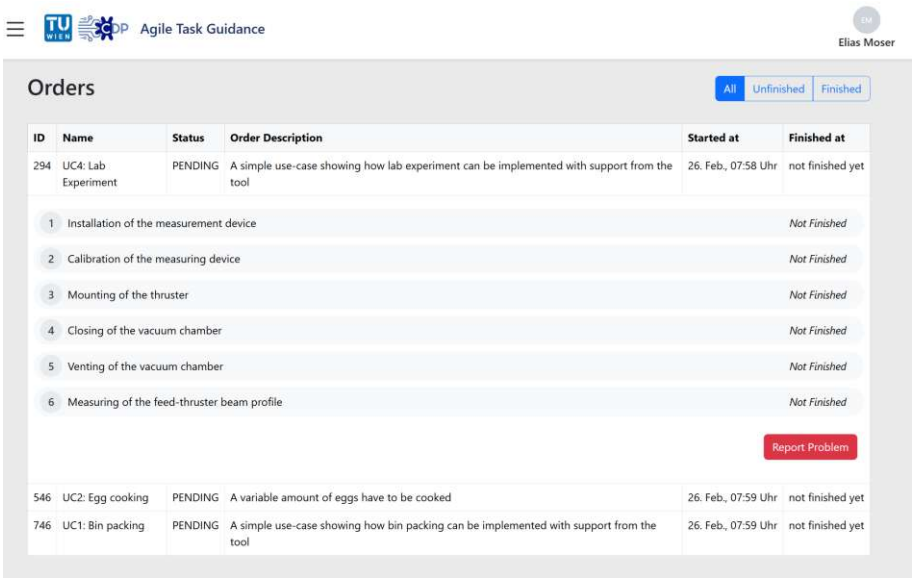


Figure 2.5: Order Overview

Once the order is imported (Figure 2.5) we can start "tasks" (Figure 2.6) and follow the workflow proposed by the solution.

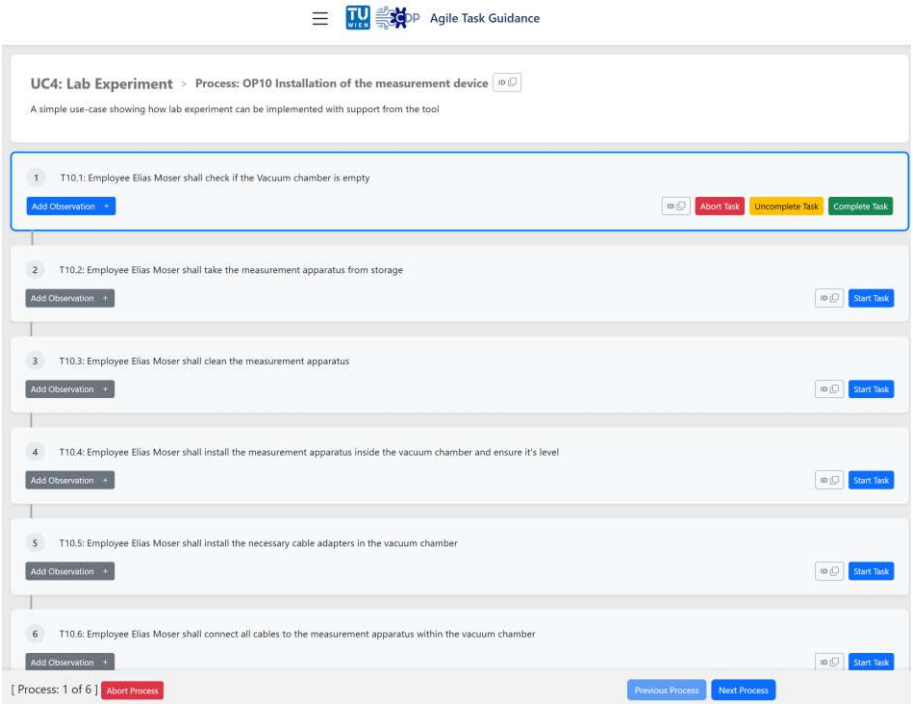


Figure 2.6: Task Detail

Even if this solution gives us a solid basis, we notice a couple of evolutions that are necessary for our work. The system is built behind three layers "Order, Phase, Task", (Figure 2.7) this lack of flexibility will be problematic when dealing with processes from the FMEA framework that can have unlimited amount of levels.

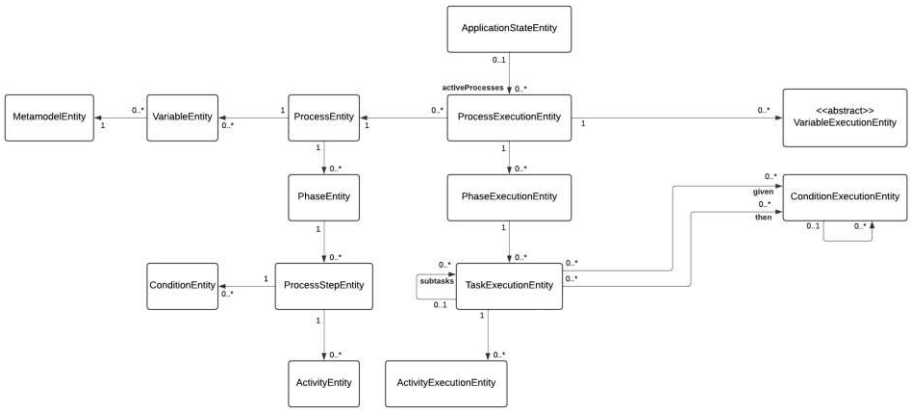


Figure 2.7: Previous Entity Diagram

The conditions for tasks to be ready for execution are rather simple and do not allow complex logic with several parameters. There is also no possibility to derive characteristics

for processes, and it is therefor impossible to include them in the conditions. Abortion and exception handling for tasks are only mocked, there is no proper way to handle errors during the execution. There is no possibility to modify processes once they are imported, which again lacks of flexibility and causes problems when changes happen during execution.

CHAPTER 3

Research Questions

3.1 Capabilities of a Process Guidance System (RQ1)

In the automotive industry, and particularly in our use case, FMEA is used as a standard to find problems and what causes them in production. But this data is mostly used by quality assurance experts to do some analysis retrospectively and usually does not give clear steps to production operators on how to actually solve these problems or stop them from happening again. There is a lot of expert knowledge available, but it is hard to make it accessible in an easy and effective way and turn it into actual ways to prevent issues. This knowledge is often stored in paper documents or digital spreadsheets, which lacks convenience, real-time support, and interactive decision making. Also, because a production plant is an ever-changing environment, when new information arises or processes change, it is difficult to add it to the existing system. We want to improve production quality, efficiency, and safety while preventing errors from returning. Before designing any solution, it is important to understand the context in depth and establish clear requirements for our artifact [34, 73]
Therefore, our first focus is on:

RQ1: What capabilities should an expert system incorporate to facilitate Process guidance for exception handling?

To answer this, we look at what other scientists have said about production exception handling in the last 10 years. We will also talk to experts from companies and research labs, to understand what they need. We perform a Literature Review [15, 41] and a Requirement Analysis [71]. This gives us a good basis for the next questions.

3.2 Prototype Implementation (RQ2)

After collecting requirements from RQ1, knowing more about the context and the needs of our work, the next step is to figure out how to build it. The current system contains details on errors but lacks step-by-step solutions. We need a way to take the FMEA data and turn it into real process instructions. With an ever-changing production line, new data continually emerges, creating the need for a system that can seamlessly incorporate this information to the existing instructions and keep everything up-to-date. The lack of version control or update rules leads to outdated or conflicting instructions. We also build a modular architecture that supports continuous improvement.
This leads to the following research question:

RQ2: What are the necessary steps and conceptual elements required for (i) creating exception handling process guidance (ii) Keeping it coherent with current production data?

We define a conceptual architecture for our solution based on the requirements we found in RQ1 and the existing solution [1]. With this done, we start implementing our artifact following agile cycles [23]. This helps us to create a meaningful prototype that actually fits the needs of the stakeholders by collecting feedback at each iteration. To ensure quality for our solution, we also take a look at the current Software Engineering best practices and follow their guidelines [73, 44, 45, 54].

3.3 Evaluation within CPPS (RQ3)

The main goal is to improve production lines and enhance their reliability. We want to prevent the same errors from happening again and keep our guidance updated. After designing and building our solution, we need to know whether it actually works and provides measurable improvements. Practical effectiveness must be proven, and improvements must be measured. The following question helps us check if our approach really improves how exceptions are handled in real production systems.

RQ3: To what extent does the proposed approach improve Exception handling in CPPS?

To address this research question, we test our APEH solution with real-world examples. This evaluation involves a Feasibility Study, Semi-Structured Interviews [55] and User Acceptance Studies [18]. This is done in a real industry setting with experts, comparing our solution with existing approaches. We also look at how much effort it takes and how efficient it is to generate and use the guidance.

CHAPTER 4

Capabilities of a Process Guidance System (RQ1)

For our first research question (RQ1), the motivation is to understand the current situation deeply in Cyber Physical Production Systems (CPPS). Although FMEA is used to identify problems and their causes, it does not provide practical, step-by-step solutions to prevent their recurrence. As a result, the same errors and inefficiencies persist, even when large amounts of data are available. We also need to understand how software in industry works by looking at systems (e.g., from our use-case). The problem is, production managers may notice recurrent issues but often lack a way to transform this data into direct, step-by-step instructions for operators. Moreover, newly incoming data make it difficult to keep guidelines up to date. It is very important to understand the needs of stakeholders like operators, production managers, and the quality control team. Operators face repeated problems, production managers need better efficiency, and the quality control team aims at high standards. Their expectations are key to developing a useful system. We must learn about key concepts of the domain, which includes FMEA, Knowledge Graphs, and Process Guidance. This knowledge supports the development of concrete scenarios and requirements for the expert system. By doing this foundational work, this represents the first draft of the expert system's capabilities, which is essential before we design the actual solution.

4.1 Methodology

To address Research Question 1 (RQ1), a two-phase approach is followed, combining a Systematic Literature Review (SLR) (Phase a) with a Scenario-Based Requirements Analysis (Phase b). This can be seen in Figure 4.1.

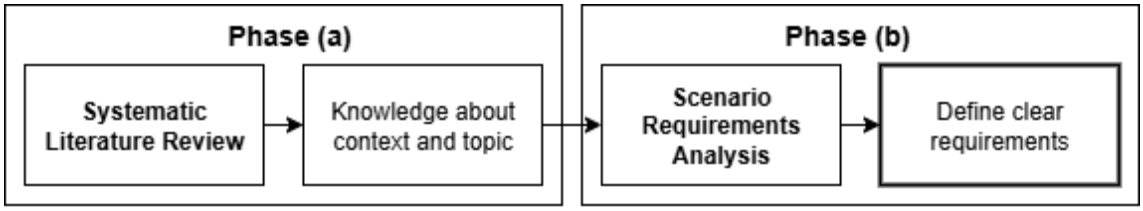


Figure 4.1: Global methodology for RQ1

4.2 Systematic Literature Review

The systematic literature review (SLR) method, consists of a structured approach to synthesize data and address research problems through transparent and reproducible steps. Carrera-Rivera's [15] & Lame's [41] papers describe these methodological steps in two main phases: planning and conducting. The planning phase includes defining a protocol, formulating focused research questions often using the PICOC (Population, Intervention, Comparison, Outcome, and Context) framework, selecting relevant digital libraries, and establishing clear inclusion and exclusion criteria. The conducting phase then involves building search strings, gathering studies from the selected databases, performing study selection and refinement (removing duplicates, screening based on criteria, and assessing quality), and finally, extracting data from the included studies.

4.2.1 Approach

Adapting the guidelines to our specific needs, we build the workflow exposed in Figure 4.2 and detail it in the following sections.

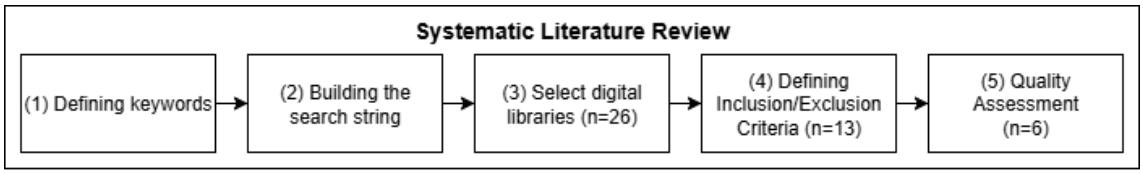


Figure 4.2: Workflow for the SLR

(1) Defining searchable keywords

The first step is to break down the SLR objectives into searchable keywords using the PICOC criteria.

- **Population** refers to the specific role, application area, or industry domain
- **Intervention** is the methodology, tool, or technology that addresses a specific issue
- **Comparison** is used to compare your intervention with another

- **Outcome** refers to the important factors for stakeholders or the expected results of the intervention

- **Context** is the setting where the comparison takes place

We summarize our keywords in Table 4.1

PICOC Element	Identified Keywords
P – Population	CPPS, Smart Factories, Manufacturing Systems, Automotive Industry, Production Errors, Exception Handling
I – Intervention	Expert Systems, Process(Task) Guidance, Knowledge Graphs, FMEA, Semantic Systems, PPR, Ontology, Operator Assistance System, Integrated Operator Decision Aid
C – Comparison	Manual Exception Handling, Traditional Methods, Checklist, Rule-based Systems, Reactive Maintenance
O – Outcome	Reduced Error Recurrence, Improved Production Efficiency, Operator Performance, Knowledge Reuse, Real-time Decision Support, System Usability, Reduced Downtime
C – Context	Industry 4.0, Agile Production Systems, Manufacturing, Automotive Production, CPPS, IoT

Table 4.1: PICOC-based Keywords for Systematic Literature Review

(2) Building the Search Strings

Here we need to build our search strings to collect papers we actually want to learn from. We combine our PICOC elements and their synonyms using boolean operators such as "OR" for synonyms and "AND" for different elements. The overall objective being to learn more about the environment and context, we are interested in several topics and try to include them all in the search string. The first part of the search string sets the context :

SubStr1 = `("manufacturing" OR "industrial production" OR "factory systems") AND ("Industry 4.0" OR "CPPS" OR "Cyber Physical Production Systems" OR "Smart Factories")`

For the second part of the string this will be adapted to each topic we are interested in and added as an "AND" close at the end of our string, this is summarized in Table 4.2.

Q1 : How do manufacturers deal with errors and exceptions? **SubStr2** = `("exception handling" OR "error handling" OR "fault recovery" OR "failure management")`
Q2 : How is process guidance used in current factories? **SubStr3** = `("process guidance" OR "task guidance" OR "Operator Assistance System" OR "Integrated Operator Decision Aid" OR "task support system")`
Q3 : How is software integrated in CPPS? **SubStr4** = `("software integration" OR "software architecture" OR "IT systems" OR "system integration")`

Table 4.2: Research Questions and Corresponding Search Strings

The final String is then : *SubStr1 AND SubStr2 AND SubStr3 AND SubStr4*

(3) Select Digital Library Sources

Our Research Question being already formulated, we switch directly to the selection of our sources. This is an important step, as the validity of your study depends on the selection of appropriate databases that cover our investigation area. Here we want to include the most meaningful publishers for scientific papers as IEEE, ACM, Springer, Wiley, Elsevier, Taylor & Francis and others. This is why we want to choose a database that indexes other publishers and will here eliminate IEEE Xplore, ACM DL, ScienceDirect and others that only references their own publication. When listing databases that indexes various sources (Google Scholar, Scopus, Web of Science) we decided to use Google Scholar that is widely use, has the broadest coverage, and is among the only ones that also includes Grey literature.

(4) Defining Inclusion and Exclusion Criteria

To narrow down our research, we will only include papers from the last 15 years that are written in English. Concerning the type of Literature we will pay more attention on academic (conference) papers but will not exclude other sources.

(5) Quality Assessment

We now have a set of thirteen papers that we can further analyze and refine. To narrow it down to around five papers that we can read, we conduct an analysis based on the abstract of each paper and evaluate if the topic fits our need or if we can tag it as out of scope. This analysis is summarized in Table 4.3

#	Paper	Assessment Result
1 [64]	The Augmented Workforce: A Systematic Review of Operator Assistance Systems, E.Roth, Handbook	Selected
2 [69]	Leveraging the Industrial Internet of Things for Business Process Improvement: A Metamodel and Patterns, C.Stoiber, Conference Paper	Selected
3 [26]	The Study of Work Design in the Industry 4.0: An Agentic Approach of Individuals and the Digital Context Interaction, L.Fréour, PhD Thesis	Selected
4	Implementation of Digital Twin in Actual Production: Intelligent Assembly Paradigm for Large-Scale Industrial Equipment, H.Ding, Conference Paper	Digital Twin – Out of scope
5	Conceptualizing Industrial IoT-based Business Process Improvements – A Metamodel and Patterns, C.Stoiber, Dissertation	Duplicate of 2
6	Research and Development of Off-line Services for the 3D Automatic Printing Machine Based on Cloud Manufacturing, C.Zhang, Conference Paper	Online 3D Printing – Out of scope
7 [80]	Learning-based Stage Verification System in Manual Assembly Scenarios, X.Zhang, Article	Selected
8	Patterns for IoT-based Business Process Improvements: Developing a Metamodel, C.Stoiber, Article	Duplicate of 2
9	Multimodal Dialogue Systems in the Era of Artificial Intelligence-Assisted Industry, V. Awasthi, Handbook	Not available
10 [70]	Improving Business Processes with the Internet of Things – A Taxonomy of IIoT Applications, C.Stoiber, Article	Selected
11	Improving Business Processes with the Internet of Things – A Taxonomy of IIoT Applications, C.Stoiber, Dissertation	Duplicate of 10
12	Exogenous Shocks and Business Process Management, M.Röglinger, Conference paper	No – Only a small interesting part
13 [33]	Scaling Up Wearable Cognitive Assistance for Assembly Tasks, R.Iyengar, PhD Thesis	Selected

Table 4.3: Quality Assessment of Selected Studies for the SLR

4.2.2 Results

We first want to take a look at the publication year and type of the remaining papers after filtering with inclusion and exclusion criteria.

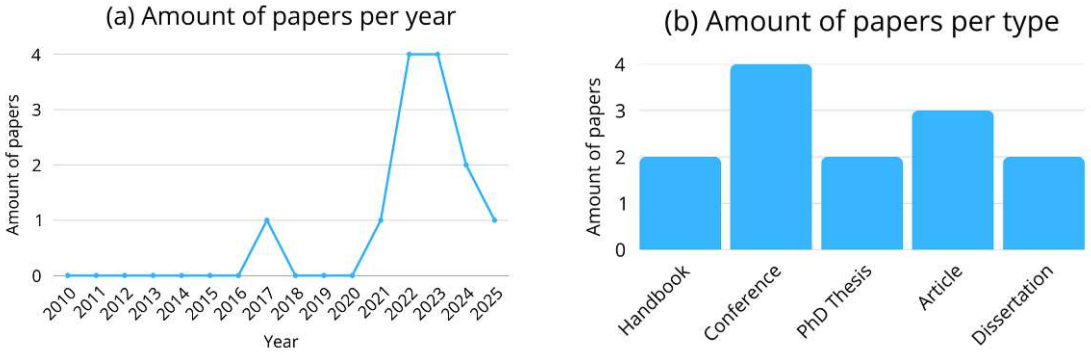


Figure 4.3: Publication year and type repartition.

On figure 4.3 (a) we can see that even filtering papers from the last 15 years, research activity was almost nonexistent before 2017, with only a single paper that year. A notable increase began in 2021, reaching peak in 2022 and 2023. This suggests that the field has gained significant attention since the early 2020s.

In figure 4.3 (b), we can see an equal repartition between the most common types of publication, which means the field is gaining interest from most of the research actors.

We now want to take a look at what the selected papers tell us about our interest areas.

How do manufacturers deal with errors and exceptions?

Traditionally, methods to handle errors and exceptions are performed by human workers. When something unexpected happens, the operator usually stops the process and checks what went wrong. Due to missing knowledge, they can rely on printed documentation or SOPs but often ask a supervisor or quality manager to double check and decide what to do [80]. If the error is not detected by the worker during production, quality control checks are done manually using checklists or predefined standards, at the end of the process. If a defect is found, the product may be sent back for rework or thrown away. This approach is based on worker experience and can sometimes lack reliability because of missing knowledge, late or missing detection. To improve those processes and build better quality products, modern factories are now starting to introduce automated systems. For example, a learning-based stage verification system [80] can monitor worker actions in real time and check the product quality using a camera. It can recognize whether the worker is doing the right task in the right order and even check angles of tools or parts. Some other systems are using wearable assistance systems to give step-by-step help to workers during their tasks. They can alert them when something goes wrong and suggest what to do next [33].

How is process guidance used in current factories?

Traditionally, process guidance is something static and paper based. Workers are initially trained for given tasks and receive printed instructions, diagrams, or charts at their station. These documents explain the assembly steps, part numbers, and tools needed. Of course, help is also provided verbally by team leaders or experienced colleagues. Some factories also use wall posters, boards, or visual signals to guide workers through different stages of the process. In more structured environments, SOP are written to define exactly how a task should be done. While effective for simple tasks, SOPs can be hard to update, especially in fast-changing production environments [64]. Recently, factories are moving toward digital process guidance, especially when product variation increases or when precision is important. Systems such as Wearable Cognitive Assistance (WCA) [33] define the task state from real time camera images to give oral feedback and images on the mobile device's screen [80]. More generally, Operator Assistance Systems (OAS) are giving step-by-step instructions mainly through visuals, using animations, 3D models, or videos [64].

How is software integrated in CPPS?

Within CPPS environments, software plays a key role by controlling machines, collecting data, and making decisions. CPPS often use the Industrial IoT, where many devices send data to software that monitors, analyzes, and reacts in real-time [70, 69]. This data is moving from devices (ex:sensors) to cloud systems or local servers in real time. The software then processes the data to detect problems, suggest improvements, or guide the next steps. This often leads to tons of data coming to the machines with the need of structure not to lose control on it. At the same time, researchers are building patterns and meta-models [69] that include best practices, common problems, and adjustable solutions to build reusable models helping to structure this mass of data. For simple collection or for "Activity Automation", the data collection done by IoT devices is nowadays mostly used to create softwares (mostly powered by AI [26]) that are used either to replace manual task or to analyze them retrospectively [69]. Some new ideas also supports human-centered design to keep the system learning and improving. This kind of cooperation between human and software is one of the goals of Industry 4.0 [26].

4.3 Scenario Requirements Analysis Method (SCRAM)

SCRAM is a method to analyze needs on a particular topic. According to Sutcliffe [71], this method describes a clear path, with four techniques to collect requirements for a solution directly with the users. First, it uses prototypes or concept demonstrators, which are simple versions of the system so users can see and give feedback. Second, it uses scenarios (or stories) showing how the system works for a user's task. This helps users understand how the prototype fits in real world. Third, design rationale is used, explaining why certain design choices were made, sometimes using a QOC (Questions, Options, Criteria) format to show different ideas and why one is chosen. This helps users

to discuss the ideas. Finally, a whiteboard summary is used during the meeting to write down the requirements and ideas together. The main steps of the method are: 1. Learn about the problem and initial needs. 2. Build the prototype, maybe simple enough just to follow a script of actions. 3. Have a session with users to show the prototype, use the scenarios, and talk about design ideas with rationale. 4. Analyze the session to get the final requirements.

4.3.1 Approach

Adapting the guidelines to our specific needs, we build the workflow exposed in Figure 4.4 and detail it in the following sections.

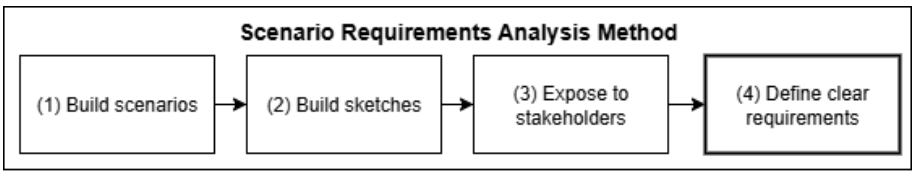


Figure 4.4: Workflow for the SCRAM

(1) Building scenarios

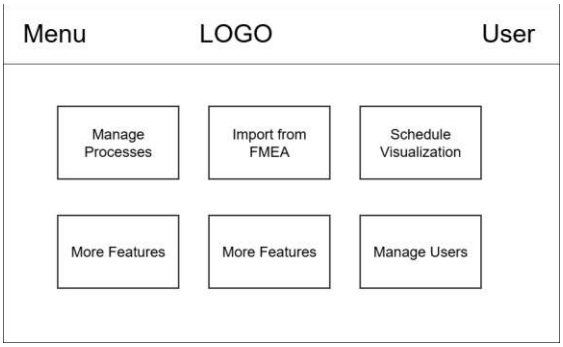
The first step of this analysis is to build scenarios of what our artifact should be capable of for specific user tasks:

1. Quality Manager imports processes from the established FMEA file.
2. The operator is starting a new manufacturing process, he is following the workflow indicated on the software by indicating what task he started, finished, and adding observations
3. The operator is having trouble during the manufacturing process, because a product characteristic is out of bound, this is not reparable, he abort the process
4. The operator is having trouble during the manufacturing process, because a product characteristic is out of bound, this being reparable, he follows the maintenance task on the software
5. The quality manager notices that maintenance tasks are outdated, he updates them on the flow
6. The quality manager wants to export the modified task to integrate them back in the FMEA file.

(2) Build sketches

For those scenarios we built a very simplified first draft of our prototype to be able to discuss it with key users. Those sketches are only for feature discussions and are not representative of the final solution. This has also been explained to our key stakeholders who participated in this analysis. You can find the sketches in Figure 4.5

(a) Process View



(b) Main Menu View

(c) Edit Process View

Figure 4.5: Simple prototype sketches for SCRAM

(3) Expose to stakeholders

With this done, we can have a session with our key stakeholders to define clear requirements for our artifact. In this meeting, we show them our prototype. We use the defined scenarios to demonstrate how it works. We also discuss our design ideas and explain our choices. This helps us getting their feedback to make sure our solution is useful and applicable. It is also a way to clarify requirements.

(4) Define clear requirements

The last step of our analysis is to define the requirements, putting all previous steps together, we manage to establish a coherent list. The result of our SCRAM is exposed in the following section.

4.3.2 Results

After exposing scenarios and prototypes to domain experts during a meeting, we managed to formulate clear requirements for this solution. We will expose them in Table 4.4.

During the session with our stakeholders, we discovered some newer details that are necessary for our solution because they are important on the field.

For a production process, a set of characteristics is defined and used for historical data management, but can also affect the workflow of the ongoing process. The management and integration of those characteristics are key to make the software valid in a real world environment.

User management is more complex than thought. The interfaces must be as simple as possible so that non-confirmed IT users can benefit from 100% of the solution. Visual standards can be used to make the interface more coherent and understandable.

Requirement ID	Description
R1 - Process Definition	
R1.1	The information system for agile process guidance shall provide the capability to import manufacturing processes from an existing FMEA file.
R1.2	The imported processes shall contain attributes (Name, Slug, Priority, ...) as well as characteristics.
R1.3	The process condition may relate to process states or to values of asset characteristics.
R2 - Process Guidance for Exception Case Handling	
R2.1	For a repairable exception case, the information system for agile process guidance shall provide the capability to guide the operator through maintenance or recovery tasks.
R2.2	For a non-repairable exception case, the information system for agile process guidance shall provide the capability for the operator to abort the process.
R3 - Process and Guidance Editing and Versioning	
R3.1	The information system for agile process guidance shall provide the capability to edit the process definition.
R3.2	The information system for agile process guidance shall provide the capability to update an outdated process and maintenance operation on-the-fly during the operation of the process.
R3.3	The information system for agile process guidance shall provide the capability for version-controlled and process modification.
R4 - Export	
R4.1	The information system for agile process guidance shall provide the capability to export modified task/process data in a format compatible with the FMEA framework.

Table 4.4: List of requirements for the APEH solution.

CHAPTER 5

Prototype Implementation (RQ2)

Now that we have a better understanding of the context of our work and managed to build, together with key stakeholders from the domain, a clear set of requirements for our solution, it is time to put this knowledge into practice and start building our artifact. Although we follow agile best practices through iteration, it is always a good idea for every software project to set the basics first. This is why before building the actual solution, we first want to define key elements for it. By key elements, we mean the architecture of the global solution together with the technical decisions about which tools we want to use. Since our solution is mostly data-driven, we also want to build a first draft of how we want our data structure to look like. This is an important step, as we will build upon that during the whole project and is challenging to change in later stages. Our overall motivation for this part is to create a bridge between theory and practice by building a concrete solution that will help us, together with our key stakeholders, to challenge the real world limitations of our system and allow us to improve it, on the fly. We want an artifact that satisfies our requirements, validates feasibility, and supports evaluation.

5.1 Research Approach

To address Research Question 2 (RQ2), we begin with an analysis of the existing solution to understand its current capabilities and limitations. Based on this analysis, the necessary improvements are identified in close alignment with the requirements gathered earlier. This allows us to build a new architectural design and implement it, following established best practices. A summary is available in Figure 5.1.

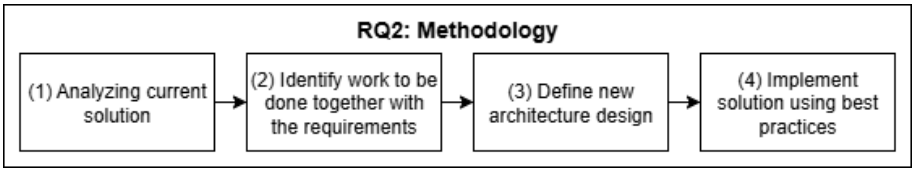


Figure 5.1: Global methodology for RQ2

5.1.1 (1) Analyze existing solution elements

Our work is based on previous work from bachelor and master students who developed a platform that allows Task Guidance as seen in section 2.5. With this project, we plan to reuse the existing solution and adapt it to the challenges and requirements of error and exception handling. In order to evaluate the changes that have to be made, it is important to understand the existing solution. It takes a lot of time and effort to learn the source code, its structure, and how the functions depend on each other. In fact, people spend 50% or more of their maintenance time just understanding the code [1].

Al-Saiyd's paper [1] describes four methodologies/models for code comprehension :

1. **Top-down :** We start with a big idea of what the program does. With this hypothesis, we break down this big idea into smaller pieces and keep doing it until you understand the program. This is used when the code is already known.

2. **Bottom-up :** Here we start by reading the actual lines of code, grouping them into bigger parts, or "chunks," that make sense together, building the knowledge step-by-step from these small parts. We go from low-level details to high-level understanding.

3. **Hybrid or Knowledge-based :** This model mixes the Top-down and Bottom-up approaches. We use our own knowledge and experience as well as the code lines and any documentation. Here it is about jumping around in the code to find links between different parts.

4. **Systematic and as-needed :** With this, maintainers only look at the code they need for a specific change.

For this work, we decided to go for a hybrid approach as this is the most time and effort efficient one. It gives us flexibility to start with a general idea of the software, but also go deep into the code details when needed. Because the code base is already quite large, This method prevents the need to read every single line of code to first gain an overall picture. For specific tasks, we can then use intuition and existing knowledge to jump to what we think are the relevant parts and explore from there, which will save us a lot of valuable time.

In addition to the code, we also used the bachelor thesis of previous students as a documentation to obtain deeper details about some parts of the software. When necessary, we also planned some meetings with them to get some insight on how they built the current solution.

Together with the requirements, this helps us to identify gaps between the existing and where we want to go. These gaps will then be transformed into work packages and artifact features.

5.1.2 Conceptual Solution Approach

(2) Requirements Driven Development

Requirements Driven Development means that all work starts from user requirements. or what users tell you they need [22]. Traditional methods write all system needs very early, like a waterfall plan, working well for products that do not change often [12].

In our work, the entire development approach is directly derived from and guided by the detailed requirements established in Chapter 4.3.2 (RQ1). However, it is important to mention that we do not follow a classical waterfall approach. We use the list of requirements as starting points to define the content of our sprints. We transform each requirement into technical specifications and design choices for the prototype. As in most software engineering projects, we use a hybrid method that combines traditional and newer methods.

Agile Methodology

Even if we defined a clear set of requirements that we want to follow to develop our artifact, we know that software projects are inherently dynamic, which makes Agile a popular way to make software now. It is different from older methods because it uses regular short release cycles, called sprints, instead of one very long cycle. This is good because it allows stakeholders to be more involved in the project. They can often see the progress of the software and can give ideas and suggestions. This helps improve the quality of the final product. When work is broken down into small tasks, it is easier to know how long it will take. By having short cycles and showing progress often, it helps people see what is happening and make suggestions [23].

For our project, we use iterative development together with recurrent meetings with our stakeholders to check the current status and validate that it is heading in the right direction. These recurrent meetings also provide opportunities to obtain constructive feedback on the fly, to continuously improve the solution and better the later evaluation.

(3) Architecture Design

To build the architecture of our system, we use the existing one and add the necessary layers. For each architectural element, we evaluate its suitability for our use case. For this,

we look at their key features, online benchmarks, popularity, and maintenance to provide arguments for our decisions. As this is a research project that is meant to be continued by other stakeholders, we also want our architecture to be easily changeable. The architecture is designed to be modular, supporting continuous improvement, scalability, and ease of maintenance, by dividing the system into smaller, independent parts.

(4) Software Engineering Best Practices

More generally through the implementation process we try to follow advices from proven research about Software Engineering Best Practices. We will not go through requirement engineering [22], modular architecture [73] and agile development [23] that we already mentioned in this section.

For the prototype development, we follow the SOLID principles from Robert C. Martin [44]. These are five important guidelines for writing better object-oriented code. They help make the system easier to understand, extend, and maintain.

- **S - Single Responsibility Principle (SRP):** Each class or function should do only one thing. We separated the code so that one part handles user input, another part handles logic, and another one deals with data. This helps to keep things simple and easier to change.
- **O - Open/Closed Principle (OCP):** Code should be open for extension but closed for modification. We designed the system in a way that allows the addition of new features without changing the main structure too much.
- **L - Liskov Substitution Principle (LSP):** This means that components should be replaceable without breaking the system. We did not use too much inheritance, but when we reused components, we made sure they behaved in the same way.
- **I - Interface Segregation Principle (ISP):** It is better to have smaller, specific interfaces than one big interface. We split responsibilities so that each module only needed the parts it actually uses.
- **D - Dependency Inversion Principle (DIP):** High-level logic should not depend on low-level details. We used configuration files and environment variables instead of writing fixed values in the code. This makes the system easier to adapt.

In addition to that, we also want to write clean code following again advices from Robert C. Martin [45] by keeping the code easy to understand and work with later, using good names for functions and variables to make them clearer, creating short functions with single purpose, using automatic tools for checking and single formatting, writing self explanatory code to avoid comments, doing clear error handling...

Since our system is used by operators and other nontechnical staff, we also follow basic principles of UI design to improve usability. These principles are based on the Norman [54] guidelines.

- **Simplicity:** The interface shows only what is necessary. We keep layouts clean and easy to understand.
- **Consistency:** We use the same colors, icons, and terms across different screens. This helps users learn the system faster.
- **Feedback:** The system gives feedback when users take actions (e.g., starting or completing a task). This let them know the system is working correctly.
- **Error Prevention:** We design the UI to reduce the chance of errors, for example, by disabling buttons when a task is not ready or showing warnings for missing input.
- **Visibility of System Status:** Users can always see in what state the process is in (e.g., task started, in progress, finished).
- **Recognition Over Recall:** We use drop-downs and buttons instead of requiring users to type or remember values.

We also pay attention to accessibility and responsive design to ensure that the system can be used in different environments.

5.2 Architecture and Implementation

In this section, we go through the whole implementation process giving step by step details about our procedure.

5.2.1 Architecture

For the architecture of our solution, we kept most of the already existing one while adapting it to our new needs. We will now go through the architecture diagram, layer by layer, that can be seen in Figure 5.2.

Infrastructure Layer (VMware)

Using the physical server provided by the CDP together with a VMware instances for virtualization, we create two virtual servers with dedicated resources. Because we decided to go for an agile project with recurrent feedback loops with our stakeholders, it is important for us to be able to deploy frequently our solution for external users. Those two servers encourage isolation by separating the development and UAT environments, allowing us to test the deployed solution on our development server before making it available to our stakeholders on the UAT server. VMware was already installed on the CDP server, we use it due to its robust virtualization capabilities, enabling resource optimization, cost-efficiency, and ease of environment replication for testing and deployment. Furthermore, it is widely recognized in industrial and enterprise environments.

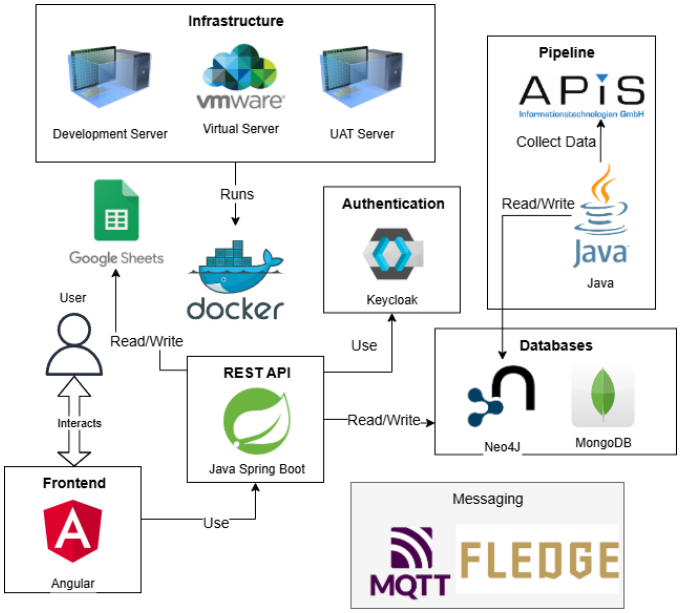


Figure 5.2: Architecture diagram of the global solution

Containerization (Docker)

With the same idea of frequently deploying our application for external users, we need a framework to make the entire solution available easily. Containerization together with Docker is nowadays a classic to build all microservices as a package and deploy them with a single docker-file. Ensuring isolation, microservices and dependencies are containerized, preventing conflicts and making debugging and deployment easier. Docker also guaranties that applications run the same across all environments reducing local environment related issues. For our infrastructure, we use VMware, but using Docker also allows us to later deploy our solution to external infrastructure for new stakeholders.

REST API (Java Spring Boot)

REST APIs follow a universally accepted architectural style for web services, using HTTP methods to manage resources. It is a very widely used standard across the software engineering domain to make backend services available in a structured way to multiple consumers. It provides a common interface between components such as GUI or third party services. It is ideal for our modular and decoupled architecture, allowing flexible and easy scaling up. To create this API, we use Java, for quite the same reasons. Java is widely used in industrial systems, offering proven reliability and long-term support. It has low programming complexity and offers efficient performance. Its worldwide use comes together with a strong community that makes rich libraries, tools, resources, and documentation easily available. Together with its Spring Boot framework, it reduces the configuration and dependency management effort.

Frontend Layer (Angular)

To communicate with the backend, we need to set up a GUI, in other words, the frontend layer. Several choices can be made among the three most widely used JavaScript frameworks (Angular, React, Vue), the initial solution was made using Angular, we kept this technology for simplicity, but also because it actually fulfills all our needs. Angular provides component-Based Architecture with modular design, encouraging re-usability, scalability, and maintainability of code. Maintained by Google it ensures recurrent updates, long term compatibility, extensive tooling, documentation, and community support.

Databases (Neo4J and MongoDB)

For the databases decisions where pretty much straightforward, Neo4J & MongoDB were used in the previous solution and the reason for their use did not change. Graph databases are commonly used in the automotive industry, where data is highly interconnected with many relationships between entities, as is the case for PPR assets. Again, Neo4J is the most common graphDB as it is optimized for complex and interconnected queries, uses the intuitive query language Cypher, and has strong integration with Java and Spring Boot. On top of that, we use MongoDB as a DocumentDB to store all information that does not take advantage of graphDB.

Authentication (Keycloak)

To provide secure and role-based access to users, the authentication layer is set using Keycloak. This is an open source single sign-on and identity management solution that is easy to set-up and has great integration with Spring Boot. It supports the most recent security standards and is updated regularly.

Google spreadsheets

The use of spreadsheets can be quite surprising in such a project whose objective is to replace old practices in terms of process exception handling. The decision to keep a small part of the solution with classical methods is due to the simplicity to define processes. We evaluated a high effort for a user to do this directly on the software and decided to keep this part external, with methods that stakeholders are already used to. This also makes it more readable and understandable for non-IT users.

FMEA Pipeline

The missing part is now the pipeline that allows us to import data from the FMEA file provided by our stakeholders directly into our software. Those files being simple HTML files, our only requirement was to be able to parse them and transform them into entities. Our solution being built on Java layers, we decided to keep that technology for the pipeline for coherence.

Messaging

This part was present in the initial solution, as it is out of scope for our work we will not detail it.

5.2.2 Data structure

Because we are building a solution on the work from previous student, we need to adapt the existing data structure (for which a simplified entity diagram can be found in Figure 2.7) to fit our new requirement.

To briefly summarize the structure of the current solution, it is composed of two parts. First, a "Definition" part (written "xxEntity" in the graph) defines a process together with all its references and dependencies. On the other hand, we have the "Execution" part (written "yyExecutionEntity" in the graph) that corresponds to the execution of a definition. It is basically a copy of the definition with assigned variables and conditions.

Each Task has a set of properties that allows to better understand and follow guidance. Their execution includes some further properties that give information about the current execution (start time, readiness, completion state, etc.). Each task also has conditions that indicate wether tasks can be executed in parallel, before or after one another.

When we import a process in the system, we create its "Definition", indicating what are its properties, phases, steps, on what variable it depends, what are the conditions linked to those processes and variables. We are not using it for the guidance itself. When we execute a process, we create an execution from the definition that assigns a value to all variables. Here, we can follow the execution state with all its properties and provide guidance.

A definition can have several executions, but an execution has a single definition. For example, we have a defined process to make pasta, we create an execution to make one kilo of pasta, but at the end of the execution, we notice that we are missing one more kilo, we then create a second execution for the same definition.

The current structure is composed of three layers (Process, Phase, Task) that are almost identical. This redundancy makes the whole solution more complex and restricts us to create multi-layered processes.

Our first objective is to remove the heavy three layered structure and modify the whole application to work with a single recursive Process (respectively ProcessExecution). Because entities are heavily dependent, this change affects almost all business services and database queries. In particular key and complex services that allow the creation of conditions and dependencies between processes as well as the evaluation done after each execution of a Task have to be entirely rethink.

In Figure 5.3, we can find a simplified version of our new entity diagram. We do not show all entities properties and related services to keep it understandable. All changes made to the previous version are indicated in green. For now we mentioned the refactoring from

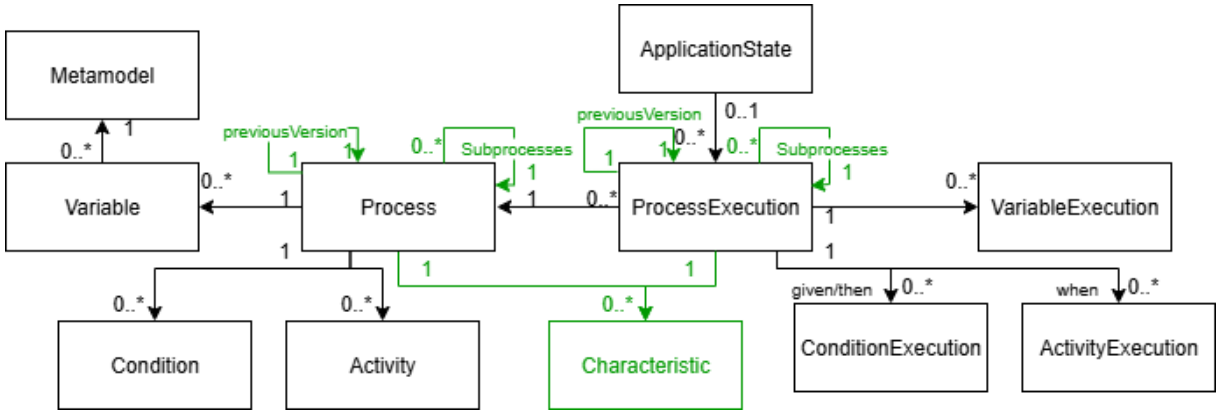


Figure 5.3: Entity diagram of the global solution

three layers to a single process layer with sub-processes, we will tackle further changes in the following sections.

5.2.3 FMEA Pipeline

We don't need to generate our own data to handle exceptions. Most of it already exists, in what we call the FMEA file. This file contains information about processes, errors that occurred, its cause, and solutions. This data is valuable for our system because we can expose it to our users when they face an issue to propose a clear way to solve it.

To make this data available in our system, we create a separated pipeline that *(1)* loads the FMEA file (in a standardized XML format) and export it into a graph format that we can use in our solution. This pipeline being a little bit complex, figure 5.4 exposes its main assets that we briefly describe below.

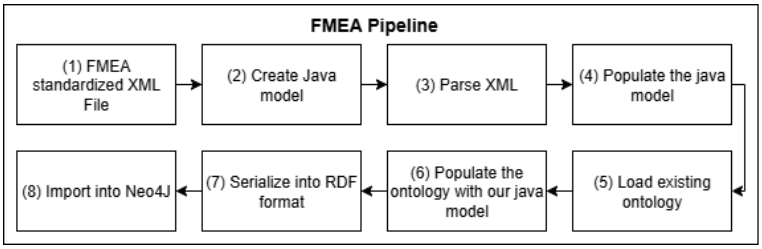


Figure 5.4: Entity diagram of the global solution

First, JAXB (Java Architecture for XML Binding) is used to define the expected structure of the XML by creating a binding context. *(2)* This context maps the elements of the XML to the corresponding Java classes, which serve as in-memory representations of the data. *(3)* To parse the XML efficiently, we use StAX (Streaming API for XML) to create an XMLStreamReader that processes the XML incrementally and allows to handle large files.

(4) Once this is done, we can use the JAXB unmarshaller that reads the StAX stream and populates the previously defined Java classes.

We then use OWLAPI to *(5)* load an existing OWL ontology. This ontology is converted into an Apache Jena Model, that provides various RDF tools to structure data in a triple format (subject-predicate-object) useful for graph modeling.

Using the populated Java classes, *(6)* the pipeline then populates the Jena model by translating the XML content into RDF triples. This includes defining resources, properties, and relationships according to the ontology's semantics. Once the model is fully constructed, Jena is used to *(7)* serialize it in RDF/XML format.

(8) The final stage consists of integrating this RDF/XML model into Neo4j. We initialize a session with the Neo4j graph database using the Neo4j Java driver. Using the n10s (Neosemantics) plugin, we load the RDF/XML content into the database.

Now that we have this data available in our system, we need to import it as a process for guidance. This will be described in the next section.

5.2.4 Import/Export Processes

Gherkin Language

In the formal system, processes are designed mostly by non-IT people that describe their definition in normal english using the gherkin language. To make this convenient for users, a google spreadsheet template is used, that we can fill out to define processes. In the spreadsheet, all processes are structured with the Gherkin syntax [1] using preconditions (Given), task activity or action (When), and postconditions (Then). Conditions can be logically chained using AND or OR operators.

For the new system, we keep this operating mode to still allow users to create processes themselves in an easily understandable way. To also keep a unique entry point to define processes in the system we will convert our FMEA data in the Gherkin Syntax and import it the same way.

Import processes

To import processes to our system, a standardized spreadsheet is used. This allows nontechnical people to easily contribute. To be able to support our new features for exception handling, a few changes had to be made to the previous template (Figure 2.4).

In the new version (Figure 5.5) we removed part of the process setup as well as the phase column that were not necessary anymore. To enable our new recursive structure, we have to mention for each process the id of its parent. Ids are voluntarily defined by the user to ensure that the same process, even if imported in two different instance of the system, keeps the same id.

[1] Cucumber. *Gherkin Reference.* Available at: `https://cucumber.io/docs/gherkin/` (Accessed: 1 August 2025).

On the right side of the sheet, we added a column to add characteristics for the process. We will go back to this new requirement later in this section but characteristics can be pretty complex, this is why we opted for a JSON format to define those characteristics. The new "isHidden" column is there to indicate if the process should be hidden from the front-end until it is ready to be executed. This is a case we need to show guidance to users only in certain case (Error case for example).

The import process itself consists of a parser service that goes through the sheet and creates all necessary nodes in our database. Here we had to adapt it to our new structure to parse processes recursively together with their characteristics. We also made a major change in the way the pre- and post- conditions were parsed. While it was possible to create simple chained conditions with AND or OR operators by creating boolean variables for each given or then component, we added a new logic to handle more complex one dealing with priorities and NOT operators.

A new requirement also mention that characteristics should be usable in our preconditions, this implies more complex logic that goes further than a boolean evaluation of variables. This problematic has been solved by adding SpEL expressions to our preconditions.

Spring Expression Language, or SpEL, is used to write small expressions in Spring applications. We can use it to get values from objects, call methods, or do basic math. It helps when we want to make things dynamic [2].

Those expressions are differentiated from variable conditions during parsing by starting with the "=" key and saved in the database as normal text. The SpEL expression is then evaluated at runtime when checking if a process precondition is fulfilled.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	**Process Setup**														
2	**Primary Actor**	kitchen_manager													
3	**Supporting Actors**	cook													
4	**Variable Definitions**	kitchen: Kitchen													
5	**Process Definition**														
6	ID	Parent ID	Slug	Name	Given	When	Then	Responsible	Priority	Duration	Notes	Handbook-URL	Is Repeatable?	Characteristics	Is Hidden?
7	1		T1	Create Dough		The worker {{ kitchen_manager }} shall create the raw dough	raw dough created	kitchen_manager	1	5m			TRUE	{ "characteristics":	FALSE
8	2	1	T1.1	Mix		The worker shall mix ingrediants	ingrediants mixed	kitchen_manager	1	5m			TRUE		FALSE
9	3		T2	Rest dough	raw dough created	The worker shall rest the dough for 30min	dough rested	kitchen_manager	1	30m			TRUE		FALSE
10	4		T3	Sheet dough	dough rested	The worker shall sheet out the dough	dough sheeted	kitchen_manager	1	5m			TRUE		FALSE
11	5		T4	Cut sheets	dough sheeted	The worker shall cut the pasta sheets		kitchen_manager	1	5m			TRUE		FALSE

Figure 5.5: New spreadsheet to import processes

Export to Gherkin

In the same way as the import feature, a new requirement impose to export the process definitions to a human easily readable format. To be consistent, we created a kind of reverse parser that converts our database entities back to Gherkin and writes them in a google spreadsheet. This allows nontechnical users to take data out of the software for modification or analysis.

[2]Spring Framework. *Spring Expression Language (SpEL)*. Available at: `https://docs.spring.io/spring-framework/docs/3.0.x/reference/expressions.html` (Accessed: 1 August 2025).

5.2.5 Process Characteristics

During our requirement collection phase, we discovered that an important part of industry processes were characteristics.

Each process can have zero, one, or more characteristics that are related. Those characteristics can be of any type (Text, Image, Video, Audio, etc.), and can have changing value over time. The value of those characteristics can also have an impact on the readiness of its process but also of other processes (e.g.: While sheeting out the pasta dough, we have a characteristic thickness, to be able to go to the next step the thickness has to be less than 5mm).

This entity does not have other relationship with the graph than being part of a process and can with time have a high number of values. This is why we decided to split the entity between two databases.

On the one hand, we have our characteristic node that can be seen as a "definition" and contains all properties for a given characteristic. When querying the graph about its processes, this allows us to have directly information about what are its characteristic without overloading the graph with a high amount of values.

On the other hand, we have characteristics values that are stored in the MongoDB as a document, here we only store information about the characteristic id, its related process id and the values itself.

As mentioned before, characteristics values can have an impact on other processes conditions and are now part of the evaluation process.

5.2.6 Exception handling

The formal system included a way to manage the conditions for each process to be ready for execution. With this we could link the completion of a process to the readiness of another with the following logic. Variables are assigned for each process pre- and post-conditions, with false value as default. When a process was successfully completed, the variable was set to true and the readiness of all processes that had this variable as precondition were re-evaluated and set to ready.

For our exception handling, we need more complex logic, as we need to invoke further processes not only in case of completion but also incompletion. More than that, even if a process is completed but one of its characteristics is not compliant, we must trigger exception handling.

For each process, we collect its characteristics in the FMEA file with its acceptable boundaries. Those are stored in our database as "normal" characteristics that have to be filled during the execution of the process. When updating the completion of this process (can be "Completed" or "Uncompleted") we evaluate all other processes that have those characteristics as precondition and check if they are in bound. If it is the case, we allow

the user to continue with normal guidance. If it is not the case, the next processes are set "unready" and the reparation processes becomes ready for execution.

In the file we also collect all causes for non-compliance, for each of those we also create a characteristic that we call "error characteristic". When such an error occurs, the user sets the task as "uncomplete", and choose to which cause it is related. We then update the corresponding error characteristic and modify the guidance as previously by showing the right reparation task.

5.2.7 Process Edition and Versioning

So far, we can define, import, and execute processes to follow guidance and deal with exceptions. This is great for systems that are not evolving so much with time, but in our case, we work in a context with permanent evolution of processes that needs more flexibility.

Because we always import our processes through the spreadsheet written in the Gherkin language, we can change the processes there and re-import them. This solution can be useful when major changes happen and no execution is in progress. In case of minor changes or processes being already in progress, we need another solution that allows modification of processes on the fly of the execution.

We noticed several cases where live modifications of processes were necessary.

- The FMEA file is humanly defined in another software, this means it is not 100% perfect and can sometimes contains incoherence or errors that are transferred to our guidance. If a user notice this incoherence while executing the guidance he will have to stop the process and report it or fix it himself. The current process cannot be simply re-import it has to be modified on the job.
- While executing a process there is a change in the production that needs modification of the guidance. Here again we need live monitoring of those processes to adapt current execution and future executions.
- For the last identified case, while trying to handle exception, we notice that the proposed solution is not efficient anymore. In this case, experience says that modification of the current process is necessary. Those modifications are sometimes really small and would demand a lot of effort to redefine the processes and re-import them into the software, live editing remains the best option.

For traceability matter, all modifications made to our processes have to be saved. This means that we need to be able to fetch at any time any past state of the current processes. This helps deciding teams to analyze changes made retrospectively by looking at the history. A new versioning system has to be designed.

At this point, we are looking for a solution to implement a versioning system that meets our requirement while being efficient and time-saving. Looking at existing solutions online that also fit our architecture, we went through a couple of them while also thinking about git that actually behaves similarly to what we wanted to achieve. Our conclusion is as follows. There is no simple existing solution to easily handle a version history graph when using Neo4J in a Java application. We have to build it ourselves and made following modification to our data structure:

- First, we add two properties to all entities related to processes. An integer version property to know how many times the entity has been modified and a boolean "isLatestVersion" property to make database access to the last version of an entity easier.
- We also add a "previousVersion" relationship to those entities to be able to navigate in the graph from one version to any older or newer ones.
- Then, we change all database queries that access those entities to filter out all entities that are not in the newer version, this ensures that everything shown by the system is in the latest state.

Now that version management is clearer, it is time to build the actual modification system. As a first important thing to keep in mind, each process has two aspects: its definition and its executions. When editing a process, we need to update both the definition but also all the executions following that definition.

Figure 5.6 helps to understand the rules that goes with this concept.

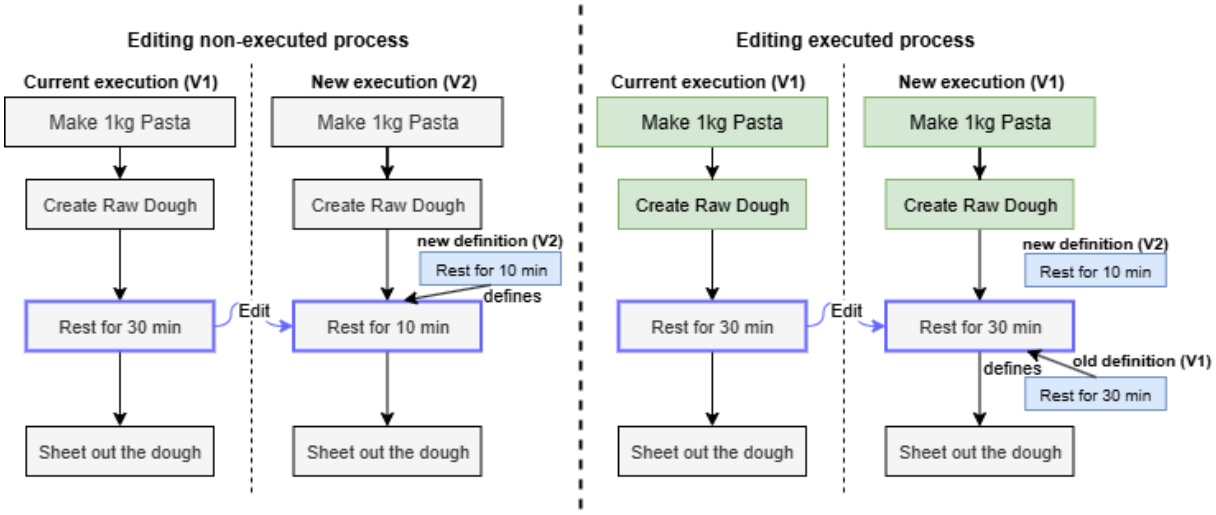


Figure 5.6: Illustration of a process modification

Here we have a defined process to make 1kg of Pasta, this process is made of 4 sub-processes. In figure 5.6 you can see the execution of this process where on the left side it did not start yet, while on the right side it started already.

We now want to edit the left side process by changing the resting time to 10 min. Because the process did not start yet, we are allowed to apply changes to this execution, this means when editing the process, we will edit both the execution and the definition.

If we now want to edit the right side process the same way, the rules changes. As we already started creating the dough, the process is under execution, we are not allowed to change the instance anymore. Here we will keep the execution as it is but still create a new definition, so that further instanced execution contains the changes. If any other execution instance having the same definition but that did not start yet exists in the software we have to edit them as well.

As you might already noticed on Figure 5.6 for each process edition we do not change the entity itself. Indeed, we create an exact copy of the current process and apply changes to this copy. While editing the copy we make sure to increment its version, tag it as latestVersion, reference the previous version, and remove the latestVersion tag from the original process. As mentioned previously if the edited process execution or one if its parent process has already started or is aborted, we do not create a copy and keep the current version as latest.

It is also important to mention that when updating a process we create a copy of its whole tree (all parents and children), because in the business, a process is seen as a collection of steps, if one of them is updated, the whole process changes.

When the modification concerns the process properties complexity stops here. But when we start altering its relationships it can become quite complex especially when we change the tree structure by updating the parent process. To better understand the issue we will detail all cases below, a graphical view can also be found in the appendix C.

The program allows the user to define loops in a process definition, in our example, the user can take the process to make 1kg of pasta and include inside a loop with 2 iterations to make 2kg of pasta. In this case, within the same tree, we have two executions of the same process definition.

We are in the case where we are altering the parent of one process.

- **Case 1:** If the process is inside a loop and we change its parent so that it stays in the loop, we are in a "normal" case.
- **Case 2:** If the process is inside such a loop and we change its parent so that it is not in the loop anymore, we are in a case where we can have several executions that needs to be removed from loops but only one that has to be added out of the loop
- **Case 3:** If the process is not inside a loop and we change its parent so that becomes in the loop, we are in a case where we have to create one process per loop iteration.

- **Case 4:** If the process is not inside a loop and we change its parent so that it stays outside the loop, we are in a "normal" case.

Because the logic behind this is quite complex, we create a pseudo-code algorithm just below.

Algorithm 5.1: Handle parent change logic – Part 1

```
1  Function HandleParentChange(processExecution):
2      if processExecution is in loop then
3          Case 1: Still in loop
4          executions ← FetchExecutions(processExecution.definition)
5          parentDef ← FetchDefinition(new parent)
6          foreach execution ∈ executions do
7              if AnyParentHasStartedOrAborted(execution) then
8                  DoNothing() ;
9              else
10                 RemoveParent(execution) ;          // also remove from
                    subprocess list
11                 parentExec ← FindExecutionInTree(parentDef)
12                 SetNewParent(execution, parentExec) ;    // also update
                    subprocess list
13             end
14         end
15     else if processExecution was in loop but now not in loop then
16         Case 2: Not in loop anymore
17         executions ← FetchExecutions(processExecution.definition)
18         parentDef ← FetchDefinition(new parent)
19         foreach execution ∈ executions do
20             if AnyParentHasStartedOrAborted(execution) then
21                 DoNothing() ;
22             else
23                 RemoveParent(execution) ;          // also remove from
                    subprocess list
24                 if new parent is not loop or is in loop then
25                     Do once:
26                     parentExec ← FindExecutionInTree(parentDef)
27                     SetNewParent(execution, parentExec)
28                 end
29             end
30         end
```

Algorithm 5.2: Handle parent change logic - Part 2 (continued)

```
1  Continuation of Function HandleParentChange(processExecution):
2      else if processExecution newly entered loop then
3          Case 3: New in the loop
4          if IsStartedOrAborted(processExecution) then
5              DoNothing() ;
6          else
7              newExec ← FetchExecutions(processExecution)
8              parentDef ← FetchDefinition(new parent)
9              RemoveParent(newExec) ;            // also remove from
                subprocess list
10             if new parent is loop or in loop then
11                 foreach loopExec that is not started or aborted do
12                     execCopy ← CopyExecution(newExec)
13                     parentExec ← parentDef in loopExec
14                     SetNewParent(execCopy, parentExec) ; // also update
                        subprocess list
15                 end
16             end
17         end
18     else if processExecution still not in loop then
19         Case 4: Still not in loop
20         if IsStartedOrAborted(processExecution) then
21             DoNothing() ;
22         else
23             newExec ← FetchExecutions(processExecution)
24             parentDef ← FetchDefinition(new parent)
25             RemoveParent(newExec) ;            // also remove from
                subprocess list
26             parentExec ← FindExecutionInTree(parentDef)
27             SetNewParent(newExec, parentExec) ;       // also update
                subprocess list
28         end
```

CHAPTER 6

Evaluation (RQ3)

As a final step, we aim to evaluate the validity of our solution. Building a prototype was already challenging, but without testing it in the real world, we cannot know if it really helps. Software is only useful if it can be effectively used and provides value in daily work. This is why we need to evaluate it in the real context of a production environment. We also seek to identify missing elements or features that do not function as expected. During development, we made many design choices. Some of them may not be fully adapted to real-life conditions. By observing how users interact with the system, it is possible to identify areas for improvement. This can also give us ideas for future work.

6.1 Feasibility Study

In this section, we first describe the approach we follow for this feasibility study, conduct it to generate insights about the software, and compare it to what was initially planned.

6.1.1 Methodology

To complete this feasibility study, we follow the methodology described in Figure 6.1. We first identify scenarios, execute them together with our APEH solution, and compare the results to the requirements set in table 4.4.

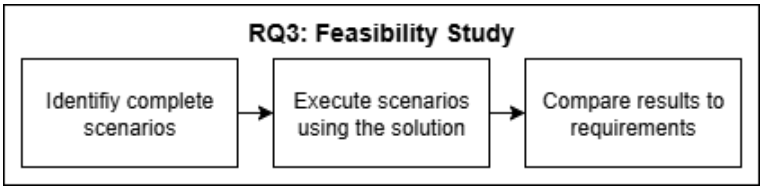


Figure 6.1: Methodology for the feasibility study

Before conducting this feasibility study, it is important to identify scenarios that we can execute with our software. These scenarios should be sufficiently comprehensive to validate the requirements set in Chapter 4. The following scenarios, derived from industry partner user stories, represent the foundation for the evaluation.

- **Use case 1: Successful execution.**
  In this scenario, Processes are imported from the FMEA file into the software. As we are only interested in part of it, a Google Spreadsheet is accessed, and unnecessary data are removed. From there we test the execution, the process is completed without any issues.

- **Use case 2: Unrepairable Error/Exception.**
  Using the same processes as in the previous use case, an issue arises during task execution. In fact, the manufactured part is too dirty and the weld seam is sprayed. As this cannot be repaired, we abort the process. The process is modified to require wearing gloves for subsequent iterations.

- **Use case 3: Repairable Error/Exception.**
  Again with the same process, another problem occurs: the weld seam is too thin, which can weaken the manufactured part. The software proposes to re-weld the part. A missing step requiring cleaning is also added. After modifying it, we export the new process to show the modifications.

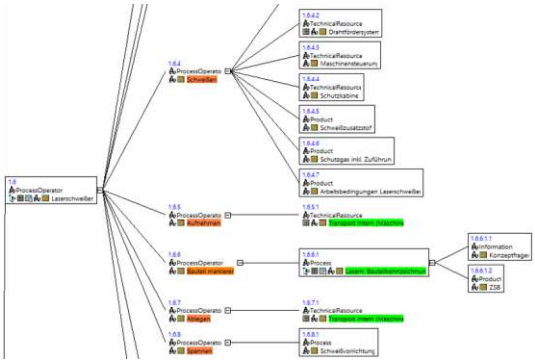
Using these scenarios, their execution can be simulated with our software and conduct evaluation by comparing our observations to the requirements.

6.1.2 Scenario execution

Due to layout issues, some pictures of this section will be shown in a small format. Those can be found in a larger format in the Appendix D

Use Case 1 : Successful execution

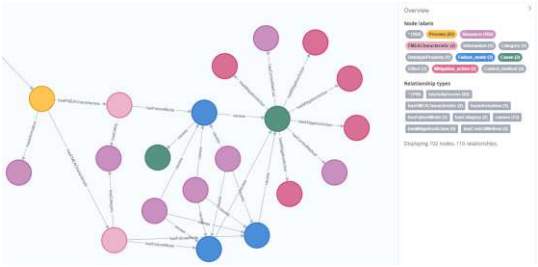
In this first scenario, we first want to fetch data available in the FMEA file.



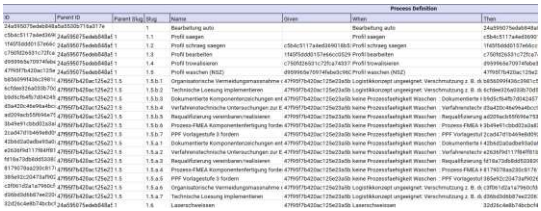
(a) First step is to collect the FMEA data in the APIS Software

(b) We then exported this data in XML format for better parsing

Figure 6.2: (a) FMEA file in the APIS software and (b) XML export of the file from APIS.



(a) With the XML export we can use our Java parser to create a graph out of it (D.1)



(b) Finally we use this graph to convert our data in Gherkin format to import it in the APEH system (D.2)

Figure 6.3: (a) Graph representation of the FMEA data in Neo4J (b) Gherkin representation of the data.

In figure 6.2 and 6.3 you can see the different steps followed to make the data available in our system. As you can see, there is a large amount of data available in the FMEA file, for better clarity, we will only take a very small part of it for our evaluation. This truncated data is now used for the rest of this section.

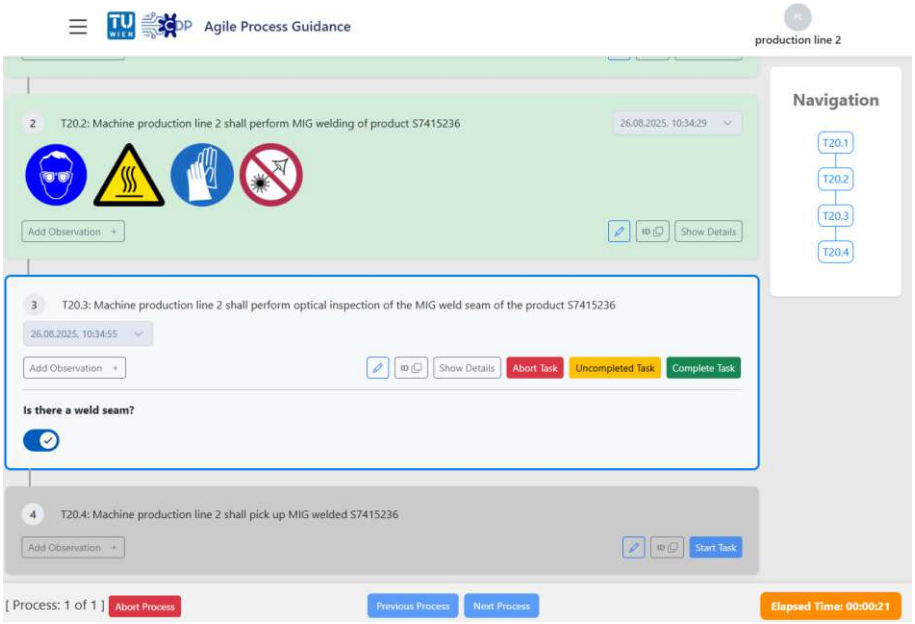


Figure 6.4: Detail process view

In figure 6.4 we can see the process detail view. We are starting all the task one after another, we see the already completed task in green, the current one in blue, and the next one in gray. On the currently started task, we can also see the green button to complete it. We cannot start the next task until we have finished the current one. On the bottom of the currently started task card, we also set our characteristic to simulate successful completion.

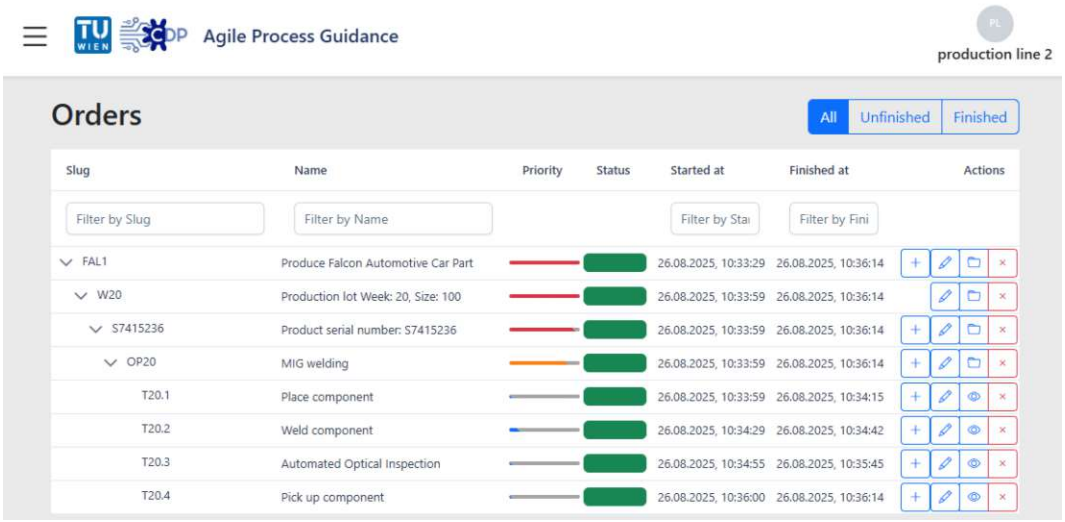


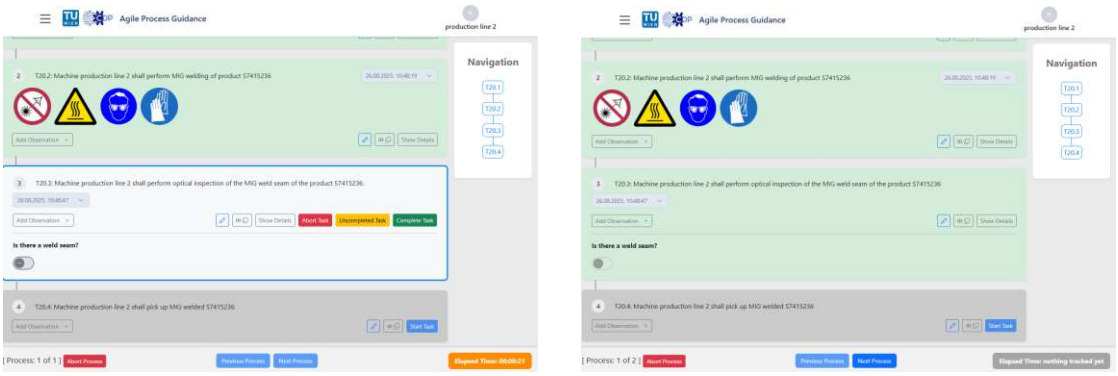
Figure 6.5: Process tree view

When all tasks are completed, we can see them in the tree view (Figure 6.5) marked green. For each process, we can show/hide its sub-processes, take a look at basic info, add a new sub-process, edit the process, show its detail going to the figure 6.4 view or abort it.

With both figures, we can see the successful completion of all processes without any issues. The characteristics having the right values we can go through the execution without issues.

Use Case 2 : Unrepairable Error

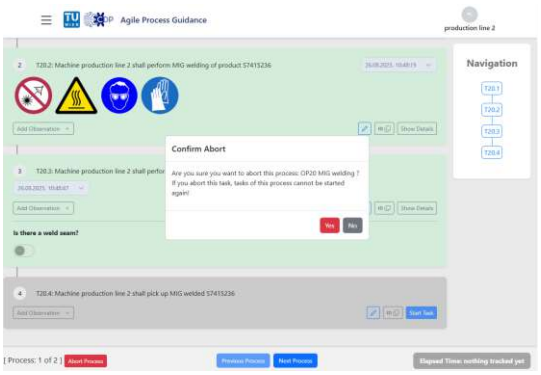
In this section we follow a path that brings to an unrepairable error, we then modify a process so that this error will not happen in the future again.



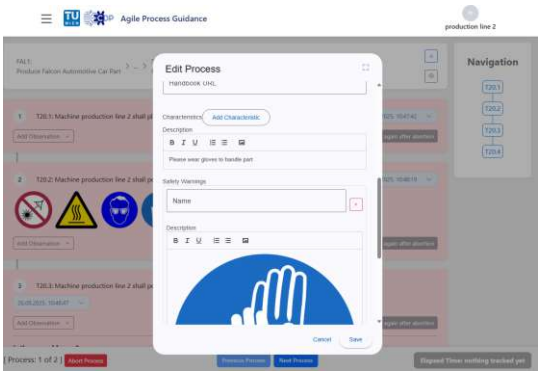
(a) We follow the process flow by indicating a problem with the weld seam

(b) Even if the process is completed we cannot go to the next step because one characteristic is in error

Figure 6.6: Execution of process with exception

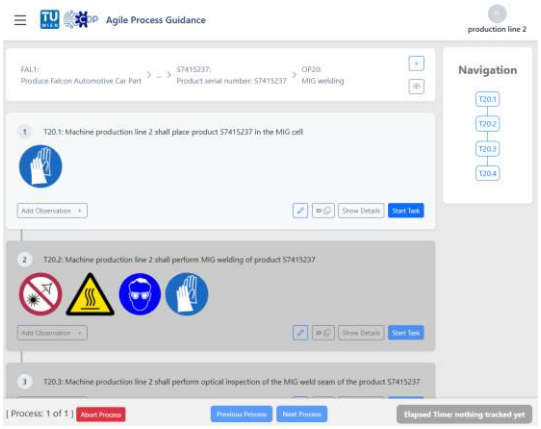


(a) Because we cannot repair this failure, we abort the process

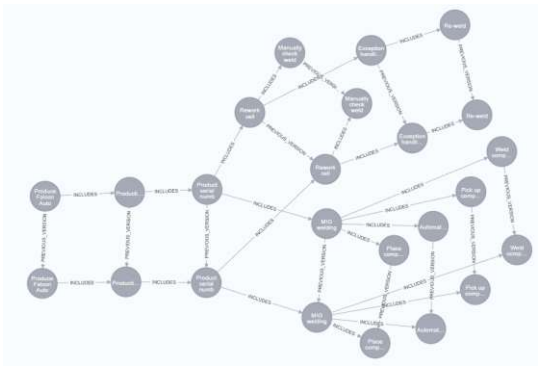


(b) After abortion we don't want this exception to happen again, we click the edit button and change the process to ask for future execution to wear gloves (D.3)

Figure 6.7: (a) Abortion popup (b) Edit Process form.



(a) On the detail view of the future execution we can now see the edited process with instruction to wear gloves to place component. The aborted execution has not been edited, we cannot change something that has been done already.(D.4)



(b) If we look at the process definitions in the graph we can see that we now have two versions. The initial version without modification and the new version with the edited process. Each process is linked to its previous version. (D.5)

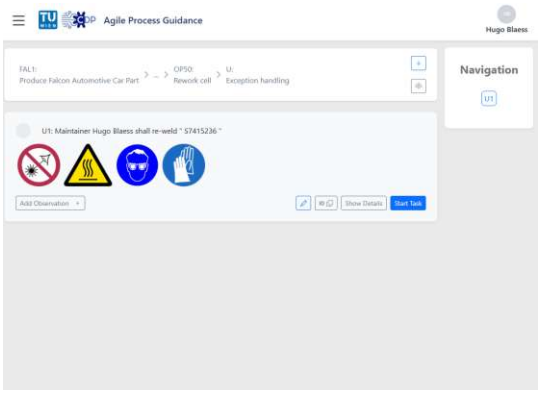
Figure 6.8: (a) Tree view with initial and edited process (b) Graph view with initial and edited process.

Figures 6.6, 6.7, and 6.8 show the full execution of our process flow when an unrepairable exception occurs. It also shows the behavior of our APEH system when we edit an aborted process.

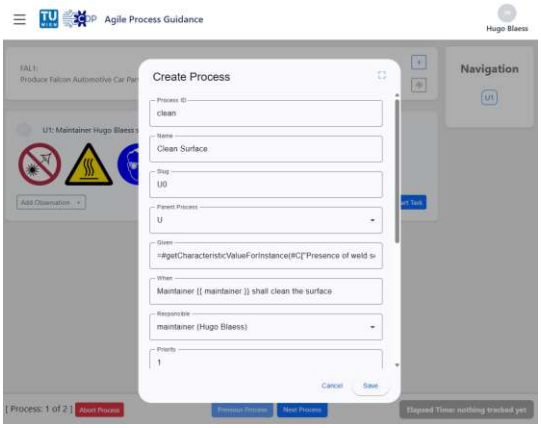
Use Case 3 : Repairable Error

In this section, we follow a path that brings to a repairable exception, we follow guidance by adding a process.

We first execute the same steps as shown in figure 6.6.

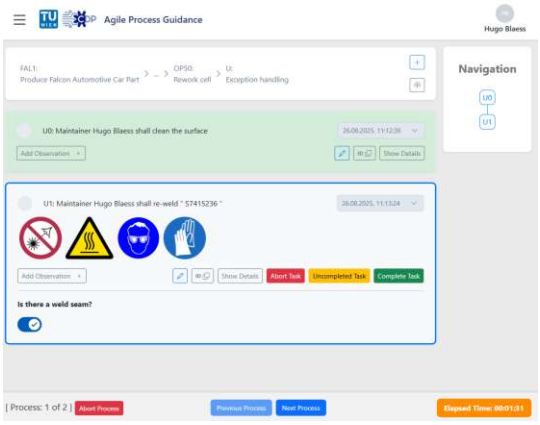


(a) Here we can see the reparation process proposed automatically by the APEH system.

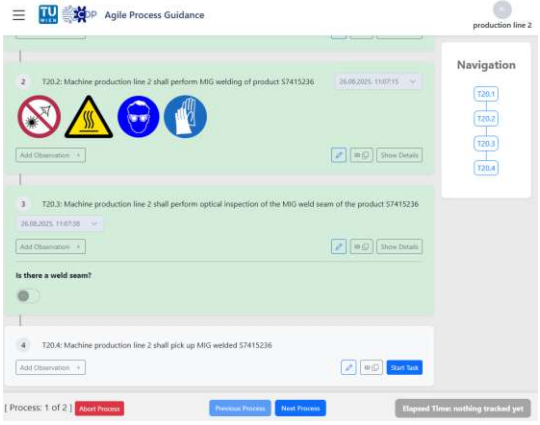


(b) By clicking the "+" button, we add a process to ask the user to clean the surface before re-welding

Figure 6.9: (a) Reparation process (b) Add process form.

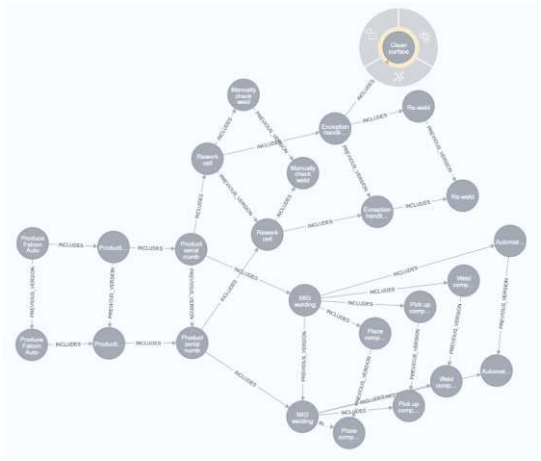


(a) With complete workflow we can now execute each process step by indicating that the weld seam is now alright (D.6)

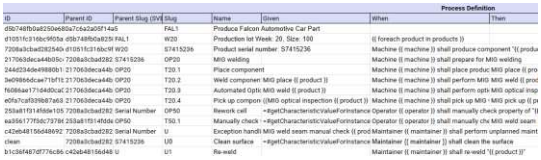


(b) Because the weld seam has been repaired we can now continue the execution of the process until the end

Figure 6.10: (a) Added process view (b) Unlocked process view.



(a) In the graph we can again see two version of the process tree. On the newest version we can see the newly added process. (D.7)



(b) By clicking the export button we are now able to see our latest modification, in gherkin language, in an google sheet. (D.8)

Figure 6.11: (a) Newly added process graph (b) Export sheet.

Figures 6.9, 6.10, and 6.11 show the full execution of our process flow when an repairable exception occurs. It also shows the behavior of our APEH system when we add a process to the current workflow.

6.1.3 Results

The results of this study are summarized in Table 6.1. The evaluation is based on following scale: "-" = Not Implemented, "+" = Partially Implemented, and "++" - Implemented. NA indicates the absence of a particular feature in the Use Case.

Requirement	UC1	UC2	UC3	Reference or explanation
R1: Process Definition				
R1.1: Import from FMEA	++	++	++	Imported FMEA in all use cases (see Figure 6.3)
R1.2: Process attributes	++	++	++	Includes name, slug, priority, and characteristics
R1.3: Conditioned execution	++	++	++	Process readiness based on status and out-of-bounds conditions
R2: Exception Handling				
R2.1: Abort on non-repairable	NA	++	NA	Abortion tested only in Use Case 2 with error classification
R2.2: Recovery for repairable	NA	NA	++	Use Case 3 includes reparation process with guided handling
R3: Process Editing and Versioning				
R3.1: Process editing	NA	++	++	Use Cases 2 and 3 edit process tree on the fly
R3.2: Live update of tasks	NA	++	++	Maintenance operations adapted live during execution
R3.3: Version control	NA	++	++	Modified processes tracked via versioning
R4: Export				
R4.1: Export to FMEA format	NA	NA	++	Export function validated in Use Case 3

Table 6.1: Requirement validation for the three use cases

6.2 User Survey

In this section, we evaluate our solution through two user oriented methods. In a first part, we describe the methods and the different steps, in a second part, we expose our results.

6.2.1 Methods

In Figure 6.12, we summarize the methodology used in this section.

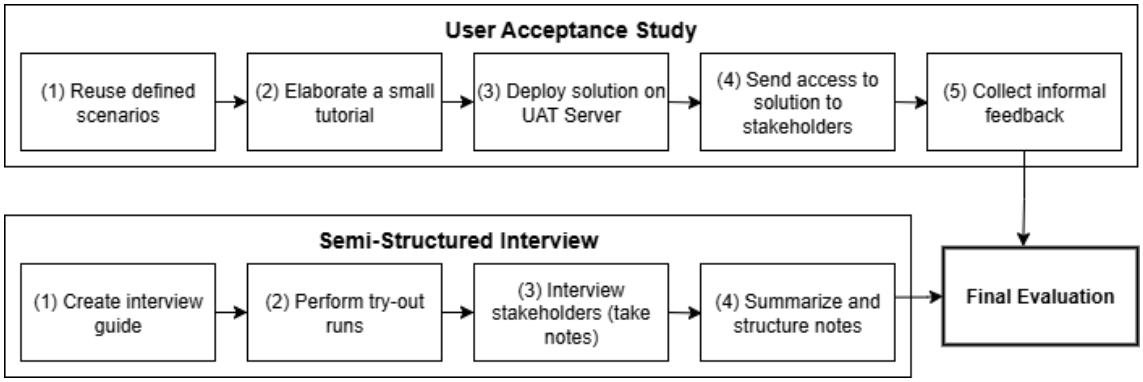


Figure 6.12: General methodology for the User Surveys.

User Acceptance Studies

For this part of the evaluation, it is all about letting the user try the software on their own without influencing the results. The objective is double, on the one side, it is about getting informal feedback from broader stakeholders about the solution. By letting people who were not involved in the development use the software on their own and then having informal discussions, we can collect valuable information about what worked well and what can still be improved. On the other side, we want to make use of this user testing to collect insights about the perceived usefulness and ease of use of stakeholders from the domain. For this, we use Davis's [18] work on the topic, especially his "Final Measurement Scales for Perceived Usefulness and Perceived Ease of Use."

The challenge is to guide users through the software without providing excessive information that could bias the results. We decided to send an email to 20 of our stakeholders from all backgrounds with the following information. We reuse the scenarios defined during the feasibility study and describe them briefly as an example of what a user can do with the software. In the same idea, to avoid revealing too much about the solution, we designed a concise tutorial of how to use the application guiding them through basic steps but letting them discover the software on their own. After deploying the solution to our UAT Server, we of course share them the link to access the online application. The last element is the measurement scale for usefulness and ease of use that we ask them to fill out when they are done testing. As a remark, we also encourage them to contact us in case of problem and to give us feedback about the solution.

Semi-Structured Interviews

Since user acceptance testing is a more autonomous task for stakeholders allowing us to gather indirect feedback from a large number of people, it is also important for us to do some more qualitative evaluation. This is the reason why we now go for semi-structured interviews, where we can reach fewer people but where we have more impact and more space to interact with our stakeholders and go deeper into the topics.

After carefully reading Nuzhat's [55] paper on the topic, we follow her guideline to structure our interviews. Conducting the interviews in a semi-structured way allows flexibility but still keep the focus on what we actually want to find out. In this way, we understand what people think and feel about our work.

According to Nuzhat, before conducting the interview we need to be sure that the format is fitting for our study. In our case they were, because they let us look at people's ideas, even if they didn't know much about the topic at first. It also allows them to talk freely, while we could still keep the conversation on track. To prepare the interview guide, we use the knowledge collected with our literature review, if needed we also ask experts in the field. The guide is a list of questions, made from the knowledge we gathered and from things we wanted to know about our software. We made sure the questions were clear and open-ended and divided them into main theme questions to start the talk and cover the main topics, and follow-up questions to ask for more details or keep us on the subject. Using "probes" that are like little hints to encourage people to share more, helped us to get deep, honest answers from everyone. The Guide can be found in Appendix E.

A very important part was testing the interview guide. We did this in a few ways. First, the research team looked at it, we acted out the interviews to find any unclear questions or if we were accidentally leading people. We also tested it with a small group of relatives who acted as study participants. This made sure the questions made sense to them and helped me know how long each interview should be. This process allowed us to make all necessary adjustments before the actual interviews.

When doing the actual interviews, we follow a set of rules. We began by briefly introducing ourselves and our research, assuring them that their answers would remain confidential, requesting permission to participate, and tell them they could stop at any time. We make sure to build a good connection with them to get honest answers. We also ask for permission to record the interviews. Recording is important because writing things down during the talk can be distracting and make you miss details. With the recordings, we can listen many times later for a detailed analysis.

The overall objective of this study is to collect metrics about usability, cognitive load, accessibility, efficiency, user empowerment, flexibility, completeness.

6.2.2 Results

User Acceptance Studies

Out of the 20 inquiries for autonomous testing that we sent, we received 13 responses, all of them filled out the questionnaire about ease of use and usefulness, for some we also received written or oral informal feedback. Among those answers, the respondents represented the following roles: Business Manager, Quality Manager, Product Manager, Domain Experts and Software Experts. In this section, we summarize the most relevant feedback, calculated the average score for each questionnaire item, analyze it, and try to build explanations for those results.

Among the informal feedback, here are some things we noticed that could be better:

- The application felt slow sometimes, due to the complex logic behind evaluation of conditions
- The system is good for small tasks but hard to use for very big processes, due to both calculation and presentation complexity.
- Lack of automation, still a lot of things have to be done by hand, more automation with other systems would be pleasant.
- Sometimes too detailed, Our software helps less experienced staff by giving step-by-step guidance. But, for expert users, this level of detail might not be needed.
- Version history can only be seen for technical users on the database. It would be valuable to integrate it into the interface for everyone.

There are also things that users liked about our software:

- The interface is clear and the guidance is easy to follow.
- It is a good way of documenting and helps to get rid of paper documents.
- It helps ensuring everyone does things the same way.
- Having version control and a history of changes is very useful.

Concerning the questionnaire, we gave a grade for each answer from 1 for extremely likely to 7 for extremely unlikely and calculated the average grade for each question, this average is shown in figure 6.13.

Looking at the results, it seems that people are not convinced that the software will help operators to be faster and more productive. In fact, for trained people this can be understandable as this just adds more bureaucracy for tasks they are already familiar with. For others, it also adds the need to report every step of the process in the software, which can be time-consuming. However, stakeholders are pretty unified on the effectiveness, usefulness, and helpfulness of the solution.

It looks like the solution is not helping operators to be faster or more productive but rather helps make their their work easier and more qualitative.

Concerning ease of use, it seems that people are comfortable with the solution. There is some adaptation to learn to operate with the software, but once the step has been taken, they find the software clear and understandable. We also noticed that people were unable to do everything they wanted because of the limited scope of the application. This opens ways to improve and add new features taking their feedback into account.

Perceived Usefulness

Using ATG in my job would enable me to accomplish tasks more quickly.

likely |_____|_____|_____|_____|_____|_____|_____| unlikely

extremely quite slightly neither slightly quite extremely

Using ATG would improve my job performance.

likely |_____|_____|_____|_____|_____|_____|_____| unlikely

extremely quite slightly neither slightly quite extremely

Using ATG in my job would increase my productivity.

likely |_____|_____|_____|_____|_____|_____|_____| unlikely

extremely quite slightly neither slightly quite extremely

Using ATG would enhance my effectiveness on the job.

likely |_____|_____|_____|_____|_____|_____|_____| unlikely

extremely quite slightly neither slightly quite extremely

Using ATG would make it easier to do my job.

likely |_____|_____|_____|_____|_____|_____|_____| unlikely

extremely quite slightly neither slightly quite extremely

I would find ATG useful in my job.

likely |_____|_____|_____|_____|_____|_____|_____| unlikely

extremely quite slightly neither slightly quite extremely

Perceived Ease of Use

Learning to operate ATG would be easy for me.

likely |_____|_____|_____|_____|_____|_____|_____| unlikely

extremely quite slightly neither slightly quite extremely

I would find it easy to get ATG to do what I want it to do.

likely |_____|_____|_____|_____|_____|_____|_____| unlikely

extremely quite slightly neither slightly quite extremely

My interaction with ATG would be clear and understandable.

likely |_____|_____|_____|_____|_____|_____|_____| unlikely

extremely quite slightly neither slightly quite extremely

I would find ATG to be flexible to interact with.

likely |_____|_____|_____|_____|_____|_____|_____| unlikely

extremely quite slightly neither slightly quite extremely

It would be easy for me to become skillful at using ATG.

likely |_____|_____|_____|_____|_____|_____|_____| unlikely

extremely quite slightly neither slightly quite extremely

I would find ATG easy to use.

likely |_____|_____|_____|_____|_____|_____|_____| unlikely

extremely quite slightly neither slightly quite extremely

Figure 6.13: Measurement Scales for Perceived Usefulness and Perceived Ease of Use adapted from Davis [18]

Semi-Structured Interviews

For the semi-structured interview we conducted a small number of interviews to ensure sufficient time for qualitative insights. The objective is also that each interviewee comes from a different background.

We conducted semi-structured interviews with six participants who work in the production field or in related roles. Each of them had experience dealing with production exceptions and using or evaluating digital tools in their work. They tested the system in simulated work conditions that imitated typical production problems. We asked them to perform example tasks using the guidance interface and gave scenarios with typical production exceptions. The goal was to understand how they see the system, how they use it, and what they think can be improved.

Participant Background

Participants have different roles and levels of experience. Two are production operators who use the system directly. One is a maintenance technician who often assists in solving technical problems. One is a quality manager responsible for analyzing failures and making sure processes follow standards. We also interviewed a researcher and a software engineer who were involved in the development and testing of the system.

Most of the participants have more than 5 years of experience. The operators explain that before using this system, they usually followed the printed SOPs or asked a colleague for help when something unexpected happened. This is similar for the technician, who mentioned that documentation is sometimes outdated or missing.

Usability and Interface

Most of the participants say that the system is easy to learn. They like the clean and not overloaded layout. Buttons and labels are clear for most users, even for operators who do not have a technical background. One participant said: "I could start using it directly, I didn't need a training session."

Still, two people said that some parts of the interface could be more visual. They suggested using more icons or color codes to make it easier to understand. One idea was to simplify the process editing screen, especially for those who are not used to working with software tools. Several people said that the use on mobile and tablet is pretty important and was neglected in the development.

Four users needed less than 30 min to get started with the software. During testing, only one of them requested help because of buttons that were too similar and caused hesitation.

Exception Handling and Decision Support

All participants agree that the system helps to improve how they react to errors. Instead of guessing or waiting for a team leader, they can follow a step-by-step guidance. An operator said: "Before, I had to call someone. Now I just follow what the screen tells me."

According to them, the system is enough to solve the task. There is no need to exit the tool to ask questions or find missing information.

Several participants said that the system is good for new workers. It gives structure and helps them act more independently. One participant said: "We don't have to explain everything every time. New operators can simply follow the guidance."

The participants said that the average time to get the team leader is around 15-20 min, and the software makes this waiting time disappear. During the test, users were also asked to edit an existing process. The researcher said: "This replaces hours of SOP changes and back-and-forth emails."

CHAPTER 7

Discussion

In this chapter, each research question is discussed separately to provide a better overview of the outcomes. We also communicate about the limitations of our system.

7.1 Capabilities of a Process Guidance System (RQ1)

Although problems in production are documented by experts in tools like FMEA, this expert knowledge usually stays locked inside PDF reports or spreadsheets. Operators benefit little from this knowledge. Even when there is good documentation about what might go wrong, there is no direct, easy-to-follow instruction for how to fix the problem. Also, current methods are too static and unsuitable for dynamic environments such as CPPS which leads to outdated instructions. This is why we need to better understand what features or capabilities a modern expert system should have to really support process guidance in such a context. To do that, we needed to understand what users really expect, what works, what does not, and what would actually make their work easier. This research question serves as a starting point for the whole work.

RQ1: What capabilities should an expert system incorporate to facilitate Process guidance for exception handling?

To answer this question, we combined two approaches. First, we did a Systematic Literature Review (SLR), to collect existing knowledge about exception handling, process guidance, and software systems in CPPS. We used the PICOC framework to define our search queries and selected papers that were meaningful and relevant to our topic. The results showed us how the industry usually reacts to exceptions, how digital guidance is slowly being introduced (e.g., AR, wearable assistants), and how software plays an increasing role in integrating systems and data. We also saw that human operators are still very important and should be supported, not replaced.

Second, we conducted a scenario-based requirement analysis (SCRAM). We created simple mock-ups and user scenarios and shared them with stakeholders. From these discussions, we were able to write down clear requirements that we summarize in Figure 4.4.

Our work aims at bringing something new and more concrete compared to what we discovered in the literature. Much of the related work focuses either on abstract models or on systems that function only in theory or in very specific settings. For example, some studies talked about using AI or knowledge graphs[20], but did not really mention how to make this usable for a worker on the production line. Others introduced Operator Assistance Systems (OAS) [64], but they often still rely on manual inputs.

We tried to combine existing ideas (FMEA, process guidance, semantic models) into one coherent and usable system. Our goal was not only to say "what would be nice to have", but to define exactly what an expert system should be able to do to help operators handle unexpected events in real time.

We focus more on flexibility and real usage. Our system is designed to work in environments that change often. It supports updates, versions, and direct edits. We also connect the process guidance to real characteristics and allow dynamic decision-making based on what is happening now, not only on what was planned before.

In general, the gap we addressed is the missing bridge between expert knowledge and operator guidance. We tried to turn static expertise into dynamic support that fits the CPPS environment and helps operators act quickly and with more confidence.

7.2 Prototype Implementation (RQ2)

After defining what the system should be able to do (RQ1), the next step was to actually try to build it. The idea was not only to create a theoretical model, but also to create a working prototype that could turn expert knowledge into usable process guidance. This was important because we needed to see how the concepts from research and requirements could be implemented in real software, and also how they behave in practice. It was also important to prepare something that real users could test and give feedback on later. This led to the following question:

RQ2: What are the necessary steps and conceptual elements required for (i) creating exception handling process guidance (ii) Keeping it coherent with current production data?

The prototype was developed according to the requirements of RQ1. We followed an agile approach, with feedback loops and continuous improvements during development. The architecture was redesigned to support flexibility, versioning, and exception handling. We introduced recursive process structures (instead of fixed layers), added support for characteristics, and created ways to handle process abortions or repairs.

We also created a data pipeline to import existing FMEA data and convert them into process definitions. This made it possible to connect expert knowledge directly to the

guidance system. A Google Sheets interface was kept for process editing, which helps involve non-technical users.

Another important part was building a versioning system. Since production processes are always changing, we made sure that processes could be edited live and older versions could be kept for traceability. This was especially important when operators noticed something wrong during execution and needed fast updates.

The gap addressed in this chapter is simply going from theoretical models and hypothesis to an actual solution as a software. We tried to build something that works with real production data, can be used by actual stakeholders, and covers both technical and usability aspects.

Compared to existing tools, our solution focuses more on agile process guidance, which can adapt over time. We also included real-time decision support, based on characteristics and user input, which goes beyond what most FMEA tools currently offer. This flexibility is also allowed by the way we handled versioning and recursive structures. The gap we addressed was not just "how to guide", but how to make the guidance system usable, editable, and scalable in real-life production lines.

7.3 Evaluation within CPPS (RQ3)

After building the prototype, we need to find out if it actually works in real situations. That's why this last research question focused on testing the solution in a CPPS environment and collecting feedback from actual users.

RQ3: To what extent does the proposed approach improves Exception handling in CPPS?

The goal was to check if the guidance helps operators during exceptions, if the software is understandable, and if the system brings some real improvement.

This step was also important to discover weaknesses or missing features that we had not thought about before. Sometimes only real usage can show what needs to be changed.

To evaluate the system, we defined real-world scenarios with successful executions, unrepairable errors, and repairable errors. We used them to simulate the execution of processes within our software.

We then observed how the system behaved by executing those test scenarios. Overall, the results were quite positive. The guidance worked as expected, the exception paths were triggered correctly. Import and export with integration of the FMEA data also worked well.

We then conducted semi-structured interviews and user acceptance tests. Feedback showed that the system is helpful and makes exception handling more transparent and easier to follow, especially for less experienced staff. Most of the users liked the idea of having step-by-step guidance directly linked to real cases and previous knowledge.

However, some limitations also became apparent. While exception handling became easier and faster, it slowed down the execution of error-free processes. When processes are very complex or involve many unknown factors, the guidance becomes less effective. Some users suggested better visual feedback.

Our evaluation tried to go further than the concept level by testing in a real CPPS context, using actual production scenarios and feedback from real users.

The main gap we tried to close is the lack of practical validation. It is not sufficient to design a system that 'could' work, it needs to be usable, understandable, and accepted by the people who are supposed to use it every day. Our evaluation showed that this is possible, even with a semi-automatic prototype, and that guidance systems can really support operators during exception handling, not just in theory but in practice.

At the same time, we also confirmed that more work is needed to make the system robust for all types of complex failures. However, the results show that our approach is a good starting point.

7.4 Limitations

Although the results of this work were promising, there are still several limitations that should be mentioned honestly.

One of the main limitations is that the solution focuses on a specific application domain and, more specifically, on a single use case. This means that the system and its features were developed for one kind of production environment, which might limit the generalization to other industries. The requirements were collected from a selected group of stakeholders who were directly involved in the use case. While this helped to focus on real needs, it also means that other perspectives might be missing.

Another limitation is the technical stack we selected. We reused some parts of an existing solution while adapting it to our needs. The benchmark of all possible tools for each architecture layer and decision based on specific criteria was out of the scope of this work. We did not consider an exhaustive list of possibilities.

The quality of the FMEA input data was another challenge. Since this data was created manually by experts using another software, it was sometimes incomplete, inconsistent, or difficult to map directly into our process structure. This caused extra work in the pipeline and also created some risks when relying too much on imported data.

Regarding the evaluation, we had a limited number of participants, and most of them had similar roles or backgrounds. This means we did not get a wide diversity of opinions. Also, the application scenarios used during the testing were selected by us and do not fully cover all possible exception situations. Some edge cases or very rare events were not tested.

We designed the interview and questionnaire in a simple way, not to overwhelm participants and limit the time we take them. This also means that some deeper insights

might have been missed. In addition, the evaluation did not take place in real production conditions, it was based on simulations and predefined scenarios. This limits how well we can judge the system under real pressure, with actual time constraints or unexpected behaviors.

Finally, the software becomes hard to use when dealing with complex processes that include many steps or loops. This leads to overloaded interfaces and a longer calculation time from the backend. These issues would need to be addressed before scaling up the system in a big factory with hundreds of processes.

CHAPTER 8

Conclusion and Future Work

8.1 Summary and Contributions

The main objective of this work was to provide real-time and contextual process guidance to support production operators during exception handling. The idea was to take the knowledge that already exists (especially in FMEA reports) and make it usable, interactive, and directly available on the production site. The goal was to reduce dependency on paper manuals or expert supervision, especially when operators face rare or unexpected situations.

To achieve this, the system needed to be simple to use, yet sufficiently intelligent and flexible enough to handle complex situations. It should allow operators to follow step-by-step guidance while also reacting to real-time data like characteristics or error causes. At the same time, the system should support engineers and quality managers in defining and updating these processes easily, without requiring too much manual effort.

Our technical goals also included: Reusing FMEA data to create workflows, enabling guidance to be edited at any time (including during execution), tracking process versions and changes, handling both repairable and non-repairable exceptions, and providing a structure that could be reused for further analysis or digital integration.

Secondary objectives included reducing defect rates, improving training efficiency, and supporting better shift handovers. These were not always easy to measure directly, but we kept them in mind throughout the development.

Finally, we wanted our system to stay aligned with real-world industry needs, so we focused on a realistic use case in the automotive sector and built the prototype in close collaboration with domain stakeholders. This ensured that the solution was not merely theoretical but also practically usable.

8.1.1 Contributions

This work contributes to the field of process guidance and exception handling in production environments in several ways. First, it proposes a semi-automatic system that connects expert knowledge with real-time operator guidance. Instead of having static paper-based instructions, our system offers dynamic workflows that can change depending on the situation, characteristics, and operator input.

We also created a modular software prototype, designed to work in evolving CPPS environments. It supports live editing, versioning, recursive process modeling, and flexible exception paths. These technical features make the system suitable for real-world applications where processes often change or need quick updates.

Lastly, the evaluation of the system with real users and realistic scenarios showed that the system is usable, understandable, and relevant for practice. It also highlighted some limitations, which were openly discussed to help future improvements.

Several communities can benefit from our work:

- **Academic Researchers:** This work provides a practical application of combining semantic knowledge graphs, agile software engineering, and human-centered design in a real manufacturing use case. It also demonstrates how design science can be applied to address both technical and social challenges in CPPS. Researchers working on smart factories, digital twins, or operator assistance can use this as a base or comparison for future work.
- **Manufacturing Operators:** The system provides a step-by-step guidance interface that replaces unclear manuals or outdated SOPs. Operators get clear instructions based on real FMEA logic and can act more independently during exception handling. This also supports less experienced staff in learning by doing, directly on the job.
- **Process Engineers:** Engineers can convert static documents into living processes that are executable and editable. They can also trace updates, handle changes through versioning, and get feedback from actual executions. This supports continuous improvement and helps reduce repeated errors on the production line.
- **Production Managers:** The system helps to increase line resilience by reacting faster to problems and reducing dependency on expert availability. Managers have better visibility over execution data, process changes, and exception causes, all of which can support decision-making and reduce production downtime or quality issues.

8.1.2 Comparison to existing solutions

Comparing our solution with existing commercial solutions described in section 2.4.1, our prototype focuses much more on exception handling and dynamic logic. While commercial

platforms usually support digital work instructions or AR-based guidance, they often rely on predefined tasks. In our case, the focus is on context-aware guidance, where processes are triggered and adapted based on conditions, characteristics, or unexpected events. We also support live editing and versioning, which is not always possible in existing tools.

Another difference is how we use existing FMEA knowledge directly as input. Most commercial systems require manual input of workflows. the prototype reuses expert input from other systems (e.g., APIS IQ) and convert it into guidance that can be executed and updated over time.

Feature	**Operations1**	**Arkite**	**VKS**	**APEH solution**
Digital Work Instructions	✓	✓	✓	✓
Exception Handling Support	×	×	×	✓
Dynamic Process Paths	✓	✓	✓	✓
FMEA Data Import	×	×	×	✓
Live Process Editing During Execution	×	×	×	✓
Process Versioning	Limited	Limited	×	✓

Table 8.1: Feature Comparison Between Commercial Solutions and the APEH solution

8.1.3 Tackled Challenges

Section 1.2 listed the main challenges faced by different stakeholders, we revisit them to summarize how our solution addresses each challenge in Table 8.2.

Challenge	**Solution Provided by the APEH System**
C1: Struggle to execute and monitor production tasks	Operators receive contextual, step-by-step guidance during exceptions, reducing dependency on manuals or experts.
C2: Difficulties to design, optimize, and update production processes	Processes are editable live, with versioning and FMEA-based workflow import.
C3: Troubles when repairing and maintaining production equipment	Failures, causes, and actions are connected through integrated FMEA logic.
C4: Can't ensure productivity and quality	All executions and exceptions are logged and can be exported for analysis and improvement.
C5: Time consuming operator training and onboarding	Guidance acts as embedded training, always up-to-date with the process.

Table 8.2: Challenges and Corresponding Solutions

8.2 Future Work

Although the evaluation showed promising results, there are still areas where the system could be improved or extended in the future.

The next logic step is to extend the functionality based on feedback from stakeholders. During the evaluation, new ideas and suggestions were shared. Some features were outside the current scope but could provide significant added value in future versions. These include interface improvements, more advanced editing options, or deeper integration with production systems.

Another potential direction is to explore application domains beyond the current automotive use case. The system was designed to be flexible and modular, so it could also be used in areas such as logistics, maintenance planning, assembly processes in electronics, or even in regulated industries like medical device manufacturing.

From a technical point of view, it would be important to improve performance, especially when scaling the system to more complex and larger processes. As seen during tests, performance decreases when many branches and recursive layers are used. Both front-end rendering and back-end calculation times could be optimized.

The system should also be tested in a real industrial environment. The current evaluation was based on realistic scenarios but not in a live production setting. A real deployment would help validate robustness, user acceptance under pressure, and the long-term benefits of the system. This would also allow us to measure the actual impact on downtime, error rates, and support load.

A further promising direction is to define and integrate key performance indicators (KPIs) to support decision-making and process control. Since the system collects feedback and execution data, it could help managers identify trends, weak points, or improvement opportunities. KPIs could include response time, exception frequency, repair success rate, etc.

Finally, future versions could include automatic measurements from machines (e.g., temperature, pressure, runtime), which would allow the system to make smarter decisions and suggest context-based actions without needing operator input for everything.

Overview of Generative AI Tools Used

While writing this thesis, we used some generative AI tools to help with language and formatting. These included **ChatGPT**[1], **Mistral**[2], and **DeepL**[3]. We used them in a limited way, just to support the writing process. We did not use them to write content or generate entire paragraphs.

Most of the time, we used ChatGPT or Mistral to improve short parts of text. For example, fixing grammar, making a sentence clearer, or checking if a formulation sounds natural. It also helped us to correctly format the LaTeX document especially with tables and figures. In every case, we reviewed the suggestions and adapted them for our work, nothing was copied from the AI.

The usual method was to take a small paragraph or a few sentences and ask the model to give feedback about mistakes, formulation problems or issues in the writing. Feedback was in the form of bullet point suggestions that we could integrate in our work manually. Following prompts were used:

For Sentence Feedback and Improvement

Can you read this paragraph and tell me if any parts sound strange or unclear? Please don't rewrite it, just give feedback or suggestions in bullet points. Text: [insert sentence or short paragraph here]

For Checking Grammar

Please list any grammar issues or awkward phrasing. Sentence: [insert sentence here]

For Refining LaTeX Expressions

Here is a section that I want to transform to a proper LaTeX syntax. Please use appropriate LaTeX tools to make it good looking in my document. Do not change any content just the presentation. Section: [insert section]

[1] ChatGPT, developed by OpenAI, is a large language model accessible at `https://chat.openai.com`

[2] Mistral is an open-weight large language model developed by Mistral AI. Model access and details available at `https://mistral.ai`

[3] DeepL is a neural machine translation service available at `https://www.deepl.com`

List of Figures

List of Tables

Glossary

APEH Agile Process Exception Handling. ix, xi, 7, 10, 32, 43, 63, 65, 68–70, 87, 93

API Application Programming Interface. 50

ATG Agile Task Guidance. 27

CDP Austrian Center for Digital Production. vii, 27, 49

CPPS Cyber Physical Production System. xiii, xiv, 2, 4, 5, 7, 11–17, 20–23, 32, 33, 35, 39, 79–82, 86, 93

FMEA Failure Mode and Effect Analysis. 2, 3, 8, 11, 13, 21–23, 29, 31–33, 35, 40, 43, 51, 53, 54, 56, 57, 64, 65, 79–82, 85–87, 91

GUI Graphical User Interface. 50, 51

HTTP Hypertext Transfer Protocol. 50

IoT Internet of Things. 35, 39

JSON JavaScript Object Notation. 55

PICOC Framework used to structure research questions, stands for Population, Intervention, Comparison, Outcome, Context. 8, 34, 35, 79, 93

PPR Product-Process-Resource. xiii, 10, 13, 17–20, 22, 35, 51

QA Quality Assurance. 2

REST REpresentational State Transfer. 50

RPN Risk Priority Number. 22, 23

SCRAM Scenario Requirements Analysis Method. 9, 39, 41, 91

SLR Systematic Literature Review. 8, 9, 33, 34, 37, 93

SOP Standard Operating Procedures. 26, 38, 39, 76, 86

SpEL Spring Expression Language. 55

UAT User Acceptance Testing. 49, 72

UI User Interface. 48, 49

Bibliography

[1] Nedhal A. Al-Saiyd. Source code comprehension analysis in software maintenance. In *2017 2nd International Conference on Computer and Communication Systems (ICCCS)*, pages 1–5, 2017.

[2] Fazel Ansari, Robert Glawar, and Tanja Nemeth. Prima: a prescriptive maintenance model for cyber-physical production systems. *International Journal of Computer Integrated Manufacturing*, 32(4-5):482–503, 2019.

[3] Fazel Ansari, Marjan Khobreh, Ulrich Seidenberg, and Wilfried Sihn. A problem-solving ontology for human-centered cyber physical production systems. *CIRP Journal of Manufacturing Science and Technology*, 22:91–106, 2018.

[4] Tim Baines, Ali Ziaee Bigdeli, Oscar Bustinza, Victor Shi, James Baldwin, and Keith Ridgway. Servitization: Revisiting the state-of-the-art and research priorities. *International Journal of Operations & Production Management*, 37:256 – 278, 02 2017.

[5] Christian Bechinie, Setareh Zafari, Lukas Kroeninger, Jaison Puthenkalam, and Manfred Tscheligi. Toward human-centered intelligent assistance system in manufacturing: challenges and potentials for operator 5.0. *Procedia Computer Science*, 232:1584–1596, 2024. 5th International Conference on Industry 4.0 and Smart Manufacturing (ISM 2023).

[6] Stefan Biffl, Detlef Gerhard, and Arndt Lüder. *Introduction to the Multi-Disciplinary Engineering for Cyber-Physical Production Systems*, pages 1–24. Springer International Publishing, Cham, 2017.

[7] Stefan Biffl, Sebastian Kropatschek, Elmar Kiesling, Kristof Meixner, and Arndt Lüder. Risk-driven derivation of operation checklists from multi-disciplinary engineering knowledge. In *2022 IEEE 20th International Conference on Industrial Informatics (INDIN)*, pages 7–14, 2022.

[8] Stefan Biffl, Sebastian Kropatschek, Kristof Meixner, David Hoffmann, and Arndt Lüder. Configuring and validating multi-aspect risk knowledge for industry 4.0 information systems. In Giancarlo Guizzardi, Flavia Santoro, Haralambos Mouratidis,

and Pnina Soffer, editors, *Advanced Information Systems Engineering*, pages 492–508, Cham, 2024. Springer Nature Switzerland.

[9] Stefan Biffl, Arndt Lüder, Kristof Meixner, Felix Rinker, Matthias Eckhart, and Dietmar Winkler. Multi-view-model risk assessment in cyber-physical production systems engineering. pages 163–170, 01 2021.

[10] Stefan Biffl, Juergen Musil, Angelika Musil, Kristof Meixner, Arndt Lüder, Felix Rinker, Danny Weyns, and Dietmar Winkler. An industry 4.0 asset-based coordination artifact for production systems engineering. In *2021 IEEE 23rd Conference on Business Informatics (CBI)*, volume 01, pages 92–101, 2021.

[11] Jon Bokrantz, Anders Skoogh, Cecilia Berlin, Thorsten Wuest, and Johan Stahre. Smart maintenance: a research agenda for industrial maintenance management. *International Journal of Production Economics*, 224:107547, 2020.

[12] J. Bosch, Helena Holmström Olsson, and Ivica Crnkovic. It takes three to tango: Requirement, outcome/data, and ai driven development. In *Workshop on Software-intensive Business*, 2018.

[13] Georg Buchgeher, David Gabauer, Jorge Martinez-Gil, and Lisa Ehrlinger. Knowledge graphs in manufacturing and production: A systematic literature review. *IEEE Access*, PP:1–1, 04 2021.

[14] Angela Carrera-Rivera, William Ochoa, Felix Larrinaga, and Ganix Lasa. How-to conduct a systematic literature review: A quick guide for computer science research. *MethodsX*, 9:101895, 2022.

[15] Angela Carrera-Rivera, William Ochoa, Felix Larrinaga, and Ganix Lasa. How-to conduct a systematic literature review: A quick guide for computer science research. *MethodsX*, 9:101895, 2022.

[16] Sonia Castelo, Joao Rulff, Erin McGowan, Bea Steers, Guande Wu, Shaoyu Chen, Iran Roman, Roque Lopez, Ethan Brewer, Chen Zhao, Jing Qian, Kyunghyun Cho, He He, Qi Sun, Vo Huy, Juan Bello, Michael Krone, and Cláudio Silva. Argus: Visualization of ai-assisted task guidance in ar. *IEEE Transactions on Visualization and Computer Graphics*, PP:1–11, 11 2023.

[17] Mathew Chacko, Atul, and Satish Babu Boddapati. A conceptual framework for cyber–physical quality monitoring system using machine learning. *Jordan Journal of Mechanical and Industrial Engineering*, 16(5):821–833, 2022. Received 1 Aug 2022; Accepted 5 Nov 2022.

[18] Fred Davis and Fred Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 13:319–, 09 1989.

[19] Barry Dowdeswell, Roopak Sinha, and Stephen G. MacDonell. Finding faults: A scoping study of fault diagnostics for industrial cyber-physical systems, 2021.

[20] Lucas Drowatzky, Martin Zinner, Kim Feldhoff, Kim Wejlupek, Jan Zimmermann, Hajo Wiemer, and Steffen Ihlenfeldt. Contributions to an fmea/fmsa based methodology to improve data quality of cyber physical production systems through digitalisation: a use case approach. 03 2025.

[21] Colin Drury. Human factors in industrial systems: 40 years on. *Human factors*, 50:368–74, 07 2008.

[22] Jinan Duan, Qunxiong Zhu, and Zhong Guan. A requirement-driven approach to enterprise application development. In Fu Lee Wang, Zhiguo Gong, Xiangfeng Luo, and Jingsheng Lei, editors, *Web Information Systems and Mining*, pages 295–302, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[23] Charles Edeki. Agile software development methodology. *European Journal of Mathematics and Computer Science*, 2(1), 2015.

[24] Emelie Engström, Margaret-Anne Storey, Per Runeson, Martin Höst, and Maria Teresa Baldassarre. How software engineering research aligns with design science: a review. *Empirical Software Engineering*, 25(4):2630–2660, July 2020.

[25] E. Francalanza, J. Borg, and C. Constantinescu. A knowledge-based tool for designing cyber physical production systems. *Computers in Industry*, 84:39–58, 2017.

[26] Léa Fréour. *The study of work design in the industry 4.0 : an agentic approach of individuals and the digital context interaction.* Theses, Université de Bordeaux ; Université libre de Bruxelles (1970-....), December 2021.

[27] Dominic Gorecky, Mathias Schmitt, Matthias Loskyll, and Detlef Zühlke. Human-machine-interaction in the industry 4.0 era. In *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, pages 289–294, 2014.

[28] Scott Gray. AIAG & VDA FMEA Handbook. Fmea handbook, Society of Collision Repair Specialists (SCAC) / AIAG & VDA, United States, September 2019. Final version presented at SCAC 2019; Copyright 2019 by AIAG and VDA.

[29] Tobias Gruber. Task scheduling in an agile task guidance system, 2024.

[30] Kagermann Henning. Recommendations for implementing the strategic initiative industrie 4.0. 2013.

[31] Béla Illés, Péter Tamás, Péter Dobos, and Róbert Skapinyecz. New challenges for quality assurance of manufacturing processes in industry 4.0. In *Precision Machining IX*, volume 261 of *Solid State Phenomena*, pages 481–486. Trans Tech Publications Ltd, 9 2017.

[32] INVET Project Team. Cyber-physical systems. `https://lms.project-invet.eu/wp-content/uploads/2018/05/6.-CYBER-PHYSICAL-SYSTEMS.pdf`, 2018. Accessed: 2025-07-09.

[33] Roger Iyengar. Scaling Up Wearable Cognitive Assistance for Assembly Tasks. 6 2023.

[34] Samireh Jalali and Claes Wohlin. Global software engineering and agile practices: A systematic review. *Journal of Software: Evolution and Process*, 24, 10 2012.

[35] MI Kellner, Ulrike Becker-Kornstaedt, William Riddle, J. Tomal, and M. Verlage. Process guides: Effective guidance for process participants. 01 1998.

[36] Matthäus Kott, Daniel Echler, and Peter Groche. Methodological approach for the development of an operator assistance system for the press shop. *The International Journal of Advanced Manufacturing Technology*, 119, 03 2022.

[37] Sebastian Kropatschek, Oskar Gert, Iman Ayatollahi, Kristof Meixner, Elmar Kiesling, Alexander Steigberger, Arndt Lüder, and Stefan Biffl. Designing a digital shadow for efficient, low-delay analysis of production quality risk. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2022.

[38] Sebastian Kropatschek, Thorsten Steuer, Elmar Kiesling, Kristof Meixner, Iman Ayatollahi, Patrik Sommer, and Stefan Biffl. Analysis of quality issues in production with multi-view coordination assets. *IFAC-PapersOnLine*, 55(10):2938–2943, 2022. 10th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2022.

[39] Christian Krupitzer, Sebastian Müller, Veronika Lesch, Marwin Züfle, Janick Edinger, Alexander Lemken, Dominik Schäfer, Samuel Kounev, and Christian Becker. A survey on human machine interaction in industry 4.0, 02 2020.

[40] Guillaume Lamé. Systematic literature reviews: An introduction. volume 1, pages 1633–1642, 07 2019.

[41] Guillaume Lamé. Systematic literature reviews: An introduction. volume 1, pages 1633–1642, 07 2019.

[42] Jay Lee, Behrad Bagheri, and Hung-An Kao. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *SME Manufacturing Letters*, 3, 12 2014.

[43] Wang Lidong and Wang Guanghui. Big data in cyber-physical systems, digital manufacturing and industry 4.0. *International Journal of Engineering and Manufacturing*, 6:1–8, 07 2016.

[44] Robert C. Martin. *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall, 2002.

[45] Robert C. Martin. *Clean code*. Addison-Wesley Professional, 1st edition edition, 2016.

[46] Kristof Meixner, Jürgen Musil, Arndt Lüder, Dietmar Winkler, and Stefan Biffl. A coordination artifact for multi-disciplinary reuse in production systems engineering. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2022.

[47] M. Moencks, E. Roth, and T. Bohné. Cyber-physical operator assistance systems in industry: Cross-hierarchical perspectives on augmenting human abilities. In *2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 419–423, 2020.

[48] Mirco Moencks, Elisa Roth, Thomas Bohné, and Per Ola Kristensson. 2022.

[49] László Monostori. Cyber-physical production systems: Roots, expectations and r&d challenges. *Procedia CIRP*, 17:9–13, 2014. Variety Management in Manufacturing.

[50] Stefan Morana, Julia Krönung, Alexander Maedche, and Silvia Schacht. Designing process guidance systems. *Journal of the Association for Information Systems*, 04 2018.

[51] Elias Moser. Operational task guidance in an agile task system, 2025.

[52] Aaron D. Neal, Richard G. Sharpe, Katherine van Lopik, James Tribe, Paul Goodall, Heinz Lugo, Diana Segura-Velandia, Paul Conway, Lisa M. Jackson, Thomas W. Jackson, and Andrew A. West. The potential of industry 4.0 cyber physical system to improve quality assurance: An automotive case study for wash monitoring of returnable transit items. *CIRP Journal of Manufacturing Science and Technology*, 32:461–475, 2021.

[53] Oliver Niggemann and Volker Lohweg. On the diagnosis of cyber-physical production systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), Mar. 2015.

[54] Don Norman. *The Design of Everyday Things*. Basic Books, revised and expanded edition, 2013.

[55] Mahnaz Aslam Nuzhat Naz, Fozia Gulab. Development of qualitative semi-structured interview guide for case study research. *Competitive Social Science Research Journal*, 3(2):42–52, Jun. 2022.

[56] Jennifer Ockerman and Amy Pritchett. A review and reappraisal of task guidance: Aiding workers in procedure following. *International Journal of Cognitive Ergonomics*, 4, 09 2000.

[57] J.J. Ockerman and A.R. Pritchett. Preliminary investigation of wearable computers for task guidance in aircraft inspection. In *Digest of Papers. Second International Symposium on Wearable Computers (Cat. No.98EX215)*, pages 33–40, 1998.

[58] João Oliveira, Gonçalo Carvalho, Bruno Cabral, and Jorge Bernardino. Failure mode and effect analysis for cyber-physical systems. *Future Internet*, 12:205, 11 2020.

[59] Shrikant Panigrahi, Khaloud Ghafri, Wafa Alyani, M. Waris, Taufiq-Hail Al-Madhagy, and Asadullah Khan. Lean manufacturing practices for operational and business performance: A pls-sem modeling analysis. *International Journal of Engineering Business Management*, 15:184797902211478, 01 2023.

[60] Günther Pawellek. *Integrierte Instandhaltung und Ersatzteillogistik: Vorgehensweisen, Methoden, Tools*. VDI-Buch. Springer Vieweg Berlin, Heidelberg, 2 edition, 2016.

[61] Zobia Rehman and Claudiu Kifor. An ontology to support semantic management of fmea knowledge. *International Journal of Computers Communications & Control*, 11:507, 07 2016.

[62] Todd A. Reichert. Applying failure modes and effects analysis (fmea) in healthcare: Preventing infant abduction, a case study. In *2004 Society for Health Systems Presentation*, representing WakeMed Healthcare System, Raleigh, NC, February 2004. Presented February 20–21, 2004.

[63] Felix Rinker, Kristof Meixner, Sebastian Kropatschek, Elmar Kiesling, and Stefan Biffl. Risk and engineering knowledge integration in cyber-physical production systems engineering. In *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 338–345, 2022.

[64] Elisa Roth, Mirco Moencks, and Thomas Bohné. *The Augmented Workforce: A Systematic Review of Operator Assistance Systems*, pages 267–279. 2023.

[65] Ioan-Matei Sarivan, Ole Madsen, and Brian Waehrens. Automatic welding-robot programming based on product-process-resource models. *The International Journal of Advanced Manufacturing Technology*, 132:1–20, 03 2024.

[66] Andreas Schumacher, Selim Erol, and Wilfried Sihn. A maturity model for assessing industry 4.0 readiness and maturity of manufacturing enterprises. *Procedia CIRP*, 52:161–166, 2016. The Sixth International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV2016).

[67] Siemens PLM Software. How to conduct a failure modes and effects analysis (fmea). White paper, Siemens PLM Software, 2016. White paper WP-60071-A3.

[68] Dieter Spath, Oliver Ganschar, Stefan Gerlach, Moritz Hämmerle, Tobias Krause, and Sebastian Schlund. Produktionsarbeit der zukunft - industrie 4.0. page 150, 06 2013.

[69] Christoph Stoiber and Stefan Schönig. Leveraging the industrial internet of things for business process improvement: A metamodel and patterns. *Information Systems and e-Business Management*, 22(2):285–313, 2024.

[70] Christoph Stoiber and Stefan Schönig. Improving business processes with the internet of things - a taxonomy of iiot applications. 05 2022.

[71] A.G. Sutcliffe and M. Ryan. Experience with scram, a scenario requirements analysis method. In *Proceedings of IEEE International Symposium on Requirements Engineering: RE '98*, pages 164–171, 1998.

[72] Thomas H.-J. Uhlemann, Christian Lehmann, and Rolf Steinhilper. The digital twin: Realizing the cyber-physical production system for industry 4.0. *Procedia CIRP*, 61:335–340, 2017. The 24th CIRP Conference on Life Cycle Engineering.

[73] Hans van Vliet. *Software Engineering: Principles and Practice*. Wiley, 2nd edition, 2008.

[74] Verein Deutscher Ingenieure (VDI) / VDE-Gesellschaft Mess- und Automatisierungstechnik. VDI/VDE 3682 Blatt 1: Formalisierte Prozessbeschreibungen – Konzept und grafische Darstellung. Richtlinie 3682 Blatt 1, VDI / VDE, Düsseldorf, Germany, May 2015. Available in German and English.

[75] Astrid von Euler. Information modelling for the manufacturing system life cycle. 2008.

[76] Sarah Wagner, Clemens Gonnermann, Marc Wegmann, Franz Listl, Gunther Reinhart, and Michael Weyrich. From framework to industrial implementation: the digital twin in process planning. *J. Intell. Manuf.*, 35(8):3793–3813, December 2023.

[77] Dietmar Winkler, Petr Novák, Kristof Meixner, Jiří Vyskočil, Felix Rinker, and Stefan Biffl. Product-process-resource asset networks as foundation for improving cpps engineering. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4, 2021.

[78] Dietmar Winkler, Serafima Sherstneva, and Stefan Biffl. Towards multi-view test specification in cpps engineering. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4, 2022.

[79] Xuan Wu, Virginie Goepp, and Ali Siadat. Cyber physical production systems: A review of design and implementation approaches. In *2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 1588–1592, 2019.

[80] Xingjian Zhang, Yutong Duan, and Zaishu Chen. Learning-based stage verification system in manual assembly scenarios, 2025.

[81] Ingo Zinnikus, Andre Antakli, Patrick Kapahnke, Matthias Klusch, Christopher Krauß, Andreas Nonnengart, and Philipp Slusallek. Integrated semantic fault analysis and worker support for cyber-physical production systems. pages 207–216, 07 2017.

Appendix

Appendix A: Vocabulary

Term	Definition
Use case	A static specification of a set of processes to be executed.
Order	The execution of a use case. It can be an order of products or an order to execute a job.
Process	An order consists of multiple processes, which can, in turn, be nested arbitrarily.
ID	An ID, is a unique identifier for an element
Task	A process at the lowest level is called a task. Tasks are the executable units.
Process instance	Processes can have multiple instances. For example, if a process fails and is restarted, these are considered separate instances.
Actor	The individual responsible for executing a specific process.
Role	A designation that defines the responsibilities and permissions. Only users assigned to a particular role can be chosen as an actor.
Resource	A Resource refers to the means used to perform a process. This can include machines, tools, human operators, software systems, ...
Product	A Product represents the result or output of one or more processes.
Characteristic	A characteristic represents a defined attribute or property that applies to a product, process, or resource.
Observation	An observation is a recorded result that is observed during one or more processes.
Slug	A Slug is a short name for the process.

Appendix B: Color Codes

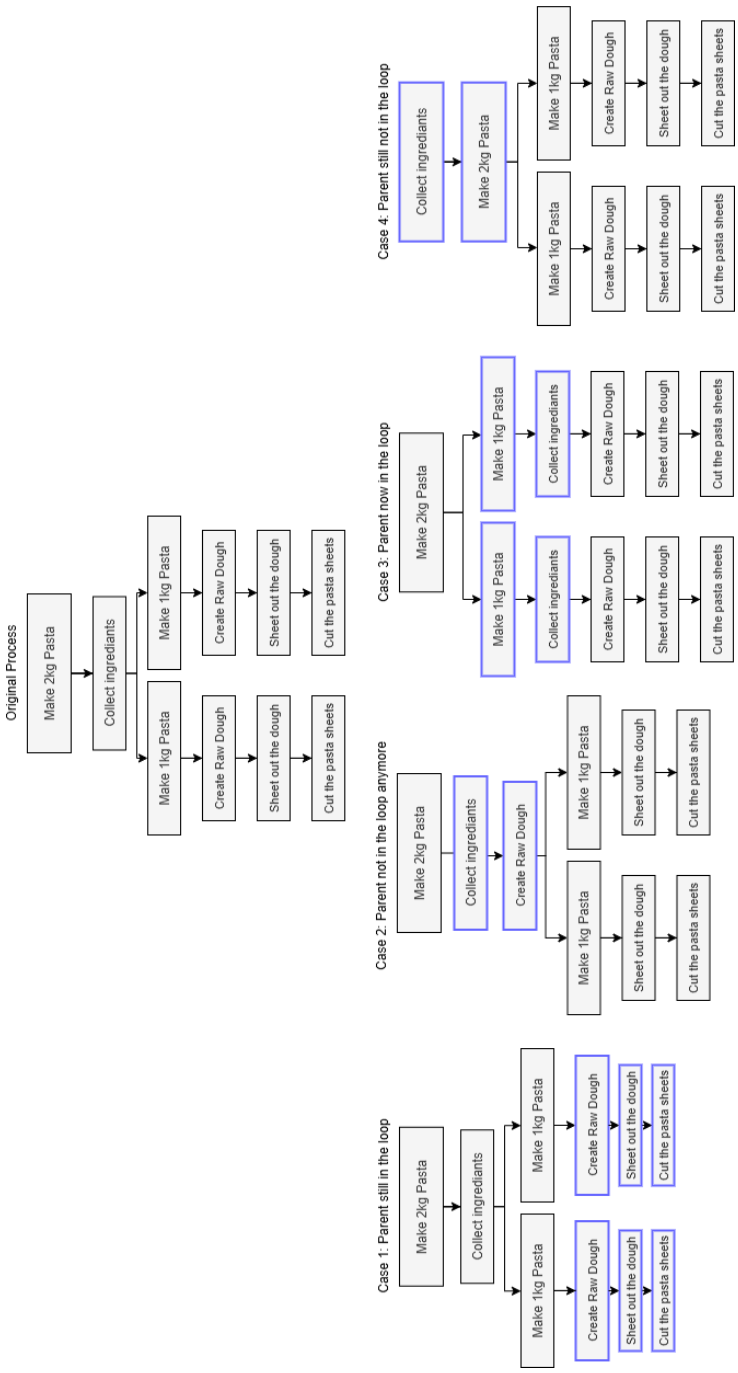
To provide a clear and immediate overview of each task's current state, the system utilizes a set of distinct colors. Each color corresponds to a specific status, enabling users to understand progress clearly and quickly identify the next steps. Table **??** lists the colors, their hex codes, and what each one means.

Color codes and their definitions

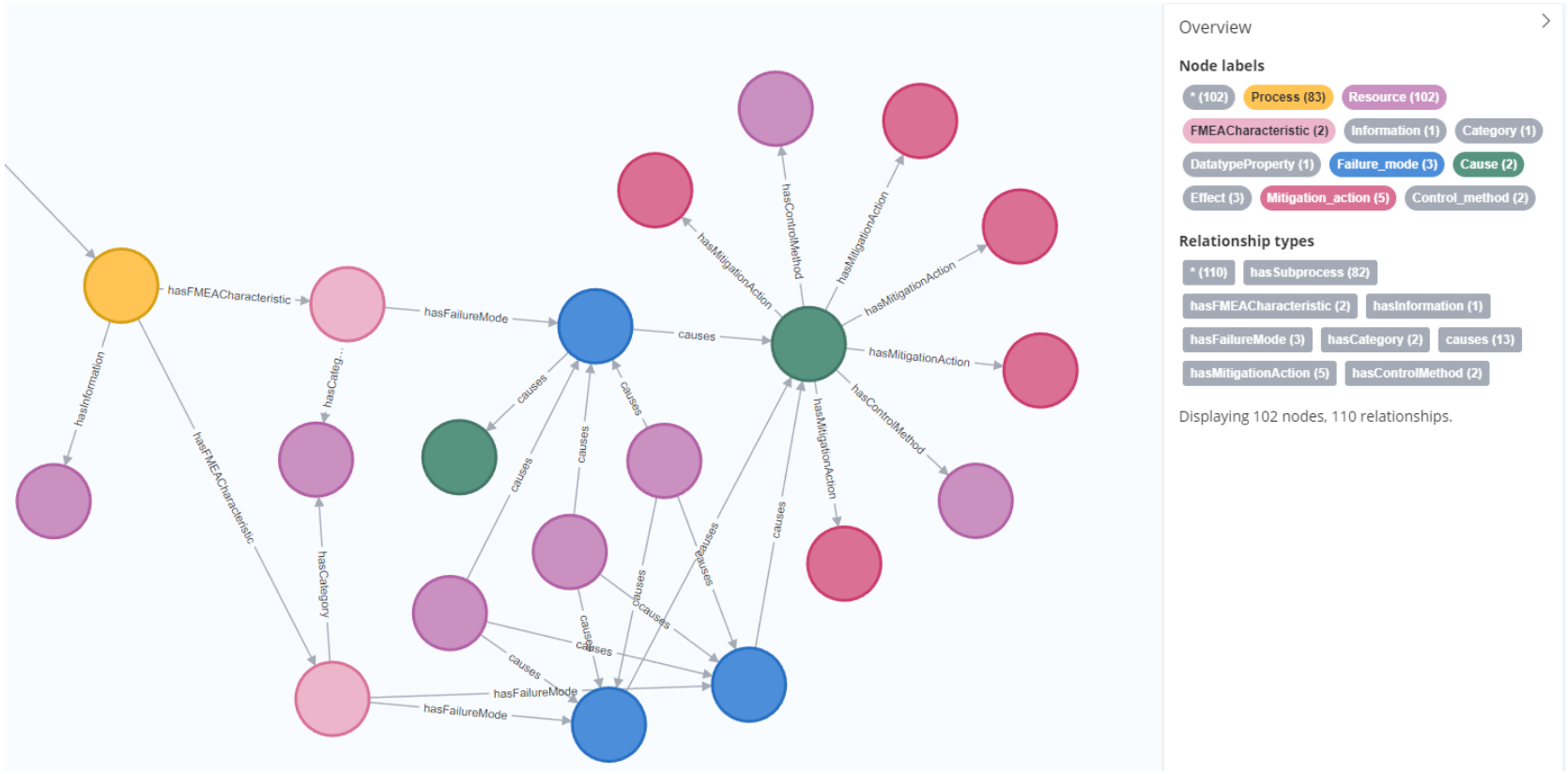
Color	Hex	Definition
Blue	#007BFF	Indicates a task is actively running
Green	#28A745	Indicates a task has finished successfully
Red	#DC3545	Indicates a task has failed or been cancelled after execution
Yellow	#FFC107	Indicates a task is uncompleted and in a warning state
White	#FFFFFF	Background color for a task that is ready to start
Dark Grey	#A5A5A5	Background color for a task that is not yet ready to be executed

These colors are applied consistently across all pages to ensure that users can recognize task states regardless of the interface context. The color coding provides a clear visual language for tracking work progress and immediately highlights tasks needing attention.

Appendix C: Usecases for parent update



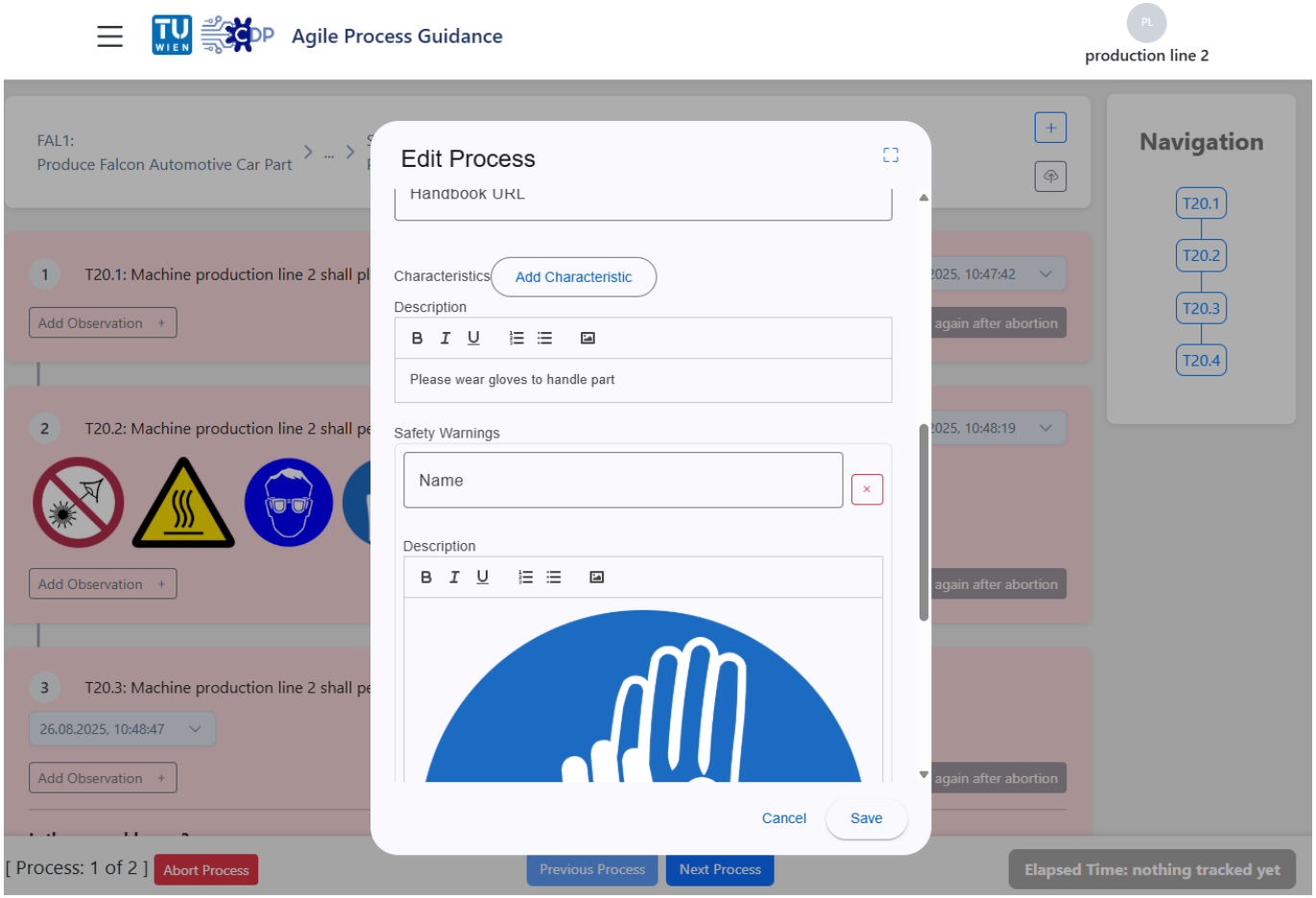
Appendix D: Evaluation pictures



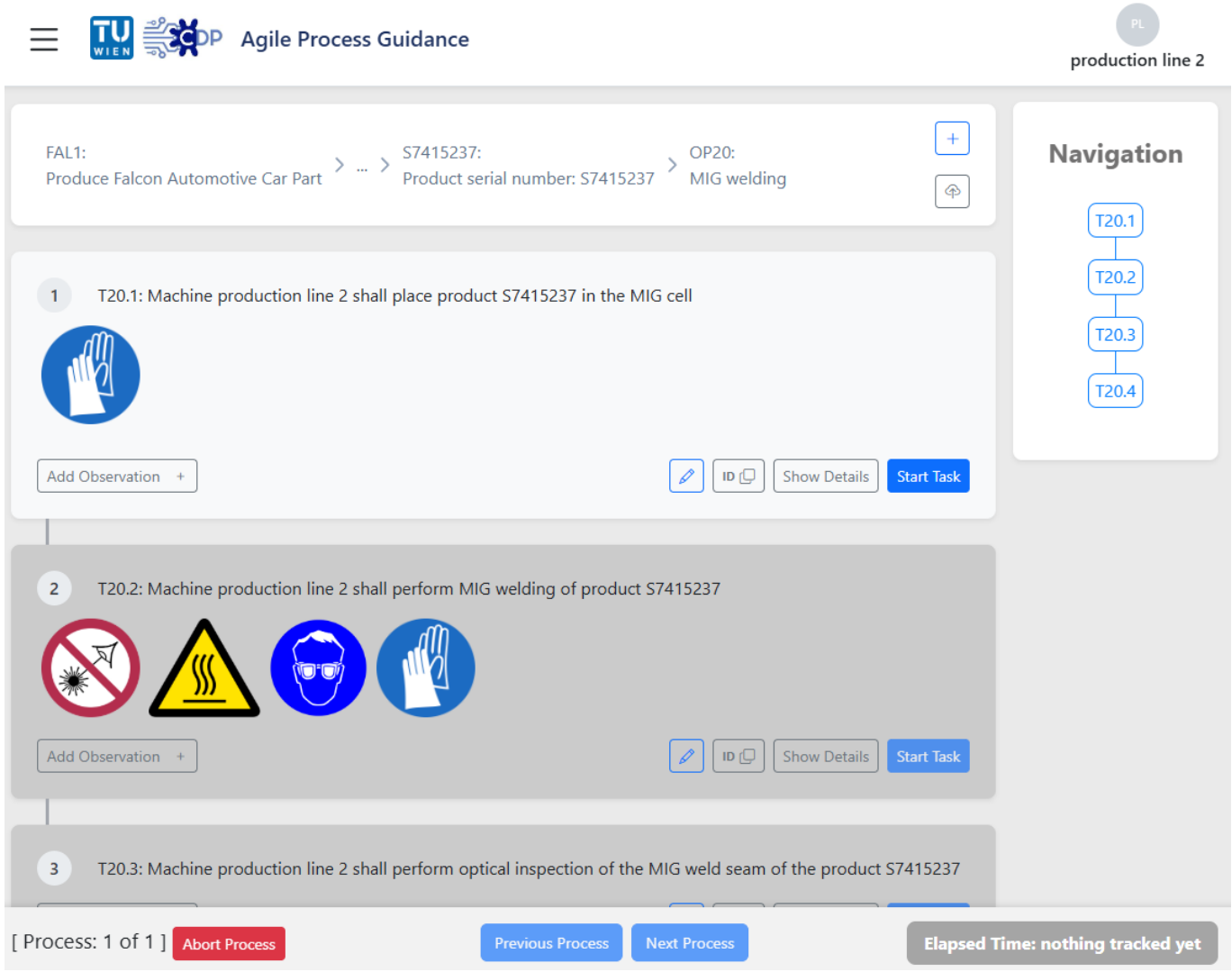
D.1: Graph representation of the FMEA data in Neo4J

						Process Definition	
ID	Parent ID	Parent Slug	Slug	Name	Given	When	Then
24a595075edeb848a	5a5530b716a317e		1	Bearbeitung auto		Bearbeitung auto	24a595075edeb848a
c5b4c5117a4ed369(	24a595075edeb848a	1	1.1	Profil saegen		Profil saegen	c5b4c5117a4ed3690
1f45f5ddd0157e66c	24a595075edeb848a	1	1.2	Profil schraeg saegen	c5b4c5117a4ed369018b5	Profil schraeg saegen	1f45f5ddd0157e66cc
c750fd26531c72fca	24a595075edeb848a	1	1.3	Profil bearbeiten	1f45f5ddd0157e66cc0529	Profil bearbeiten	c750fd26531c72fca7
d959965e70974febe	24a595075edeb848a	1	1.4	Profil trowalisieren	c750fd26531c72fca74337	Profil trowalisieren	d959965e70974febe3
47f95f7b420ac125e	24a595075edeb848a	1	1.5	Profil waschen (NSZ)	d959965e70974febe3c960	Profil waschen (NSZ)	47f95f7b420ac125e2
b856099f436c3981c	47f95f7b420ac125e23	1.5	1.5.b.1	Organisatorische Vermeidungsmassnahme c	47f95f7b420ac125e23a5b	Logistikkonzept ungeeignet: Verschmutzung z. B. du	b856099f436c3981c5
6cfdee326a033b70d	47f95f7b420ac125e23	1.5	1.5.b.2	Technische Loesung implementieren	47f95f7b420ac125e23a5b	Logistikkonzept ungeeignet: Verschmutzung z. B. du	6cfdee326a033b70d5
b9d5cf64fb7d04245	47f95f7b420ac125e23	1.5	1.5.b.3	Dokumentierte Komponentenzeichungen ent	47f95f7b420ac125e23a5b	keine Prozessfaehigkeit Waschen : Dokumentierte K	b9d5cf64fb7d042457
d3a420c46e96a4bc(	47f95f7b420ac125e23	1.5	1.5.b.4	Verfahrenstechnische Untersuchungen zur E	47f95f7b420ac125e23a5b	keine Prozessfaehigkeit Waschen : Verfahrenstechr	d3a420c46e96a4bcc9
ad209acb55f696e75	47f95f7b420ac125e23	1.5	1.5.b.5	Requalifizierung vereinbaren/realisieren	47f95f7b420ac125e23a5b	keine Prozessfaehigkeit Waschen : Requalifizierung	ad209acb55f696e753
3b49e91cbbd02a3a(	47f95f7b420ac125e23	1.5	1.5.b.6	Prozess-FMEA Komponentenfertigung forde	47f95f7b420ac125e23a5b	keine Prozessfaehigkeit Waschen : Prozess-FMEA K	3b49e91cbbd02a3a82
2cad47d1b469e8d0(	47f95f7b420ac125e23	1.5	1.5.b.7	PPF Vorlagestufe 3 fordern	47f95f7b420ac125e23a5b	keine Prozessfaehigkeit Waschen : PPF Vorlagestuf	2cad47d1b469e8d092
42b6d2a0adbe93a0(	47f95f7b420ac125e23	1.5	1.5.a.1	Dokumentierte Komponentenzeichungen ent	47f95f7b420ac125e23a5b	keine Prozessfaehigkeit Waschen : Dokumentierte K	42b6d2a0adbe93a0a(
e2636f9d117f84ff81	47f95f7b420ac125e23	1.5	1.5.a.2	Verfahrenstechnische Untersuchungen zur E	47f95f7b420ac125e23a5b	keine Prozessfaehigkeit Waschen : Verfahrenstechr	e2636f9d117f84ff81b
fd18a73db8dd5338(	47f95f7b420ac125e23	1.5	1.5.a.3	Requalifizierung vereinbaren/realisieren	47f95f7b420ac125e23a5b	keine Prozessfaehigkeit Waschen : Requalifizierung	fd18a73db8dd533839
8179078aa230c817(	47f95f7b420ac125e23	1.5	1.5.a.4	Prozess-FMEA Komponentenfertigung forde	47f95f7b420ac125e23a5b	keine Prozessfaehigkeit Waschen : Prozess-FMEA K	8179078aa230c8176
385e92c20473af902	47f95f7b420ac125e23	1.5	1.5.a.5	PPF Vorlagestufe 3 fordern	47f95f7b420ac125e23a5b	keine Prozessfaehigkeit Waschen : PPF Vorlagestuf	385e92c20473af9026
c3f061d2a1a7960cf	47f95f7b420ac125e23	1.5	1.5.a.6	Organisatorische Vermeidungsmassnahme c	47f95f7b420ac125e23a5b	Logistikkonzept ungeeignet: Verschmutzung z. B. du	c3f061d2a1a7960cfd
d36bd3d6b87ee220(	47f95f7b420ac125e23	1.5	1.5.a.7	Technische Loesung implementieren	47f95f7b420ac125e23a5b	Logistikkonzept ungeeignet: Verschmutzung z. B. du	d36bd3d6b87ee2206
32d26c4e8b74bcbcf	24a595075edeb848a	1	1.6	Laserschweissen	47f95f7b420ac125e23a5b	Laserschweissen	32d26c4e8b74bcbcf4

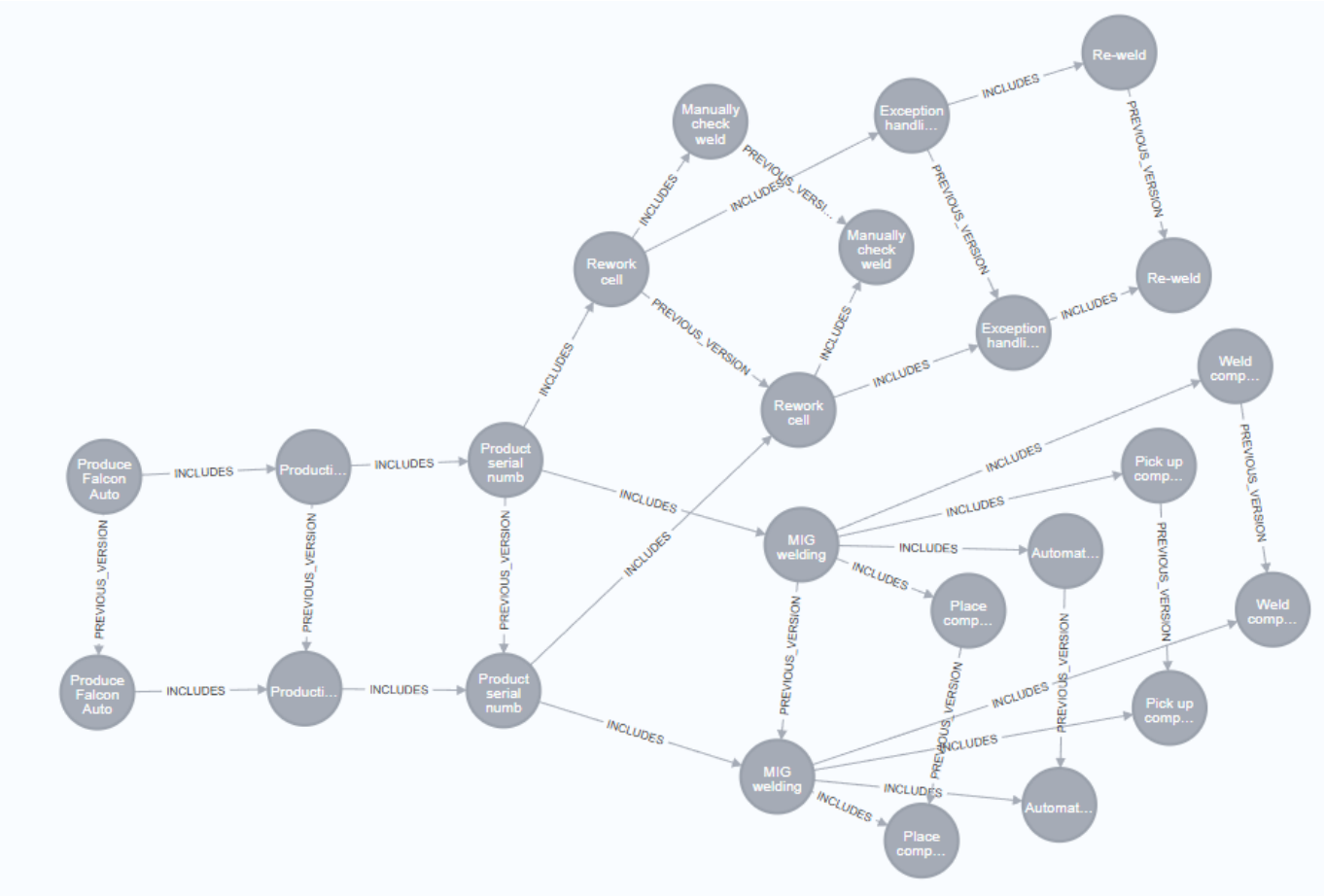
D.2: Gherkin representation of the data



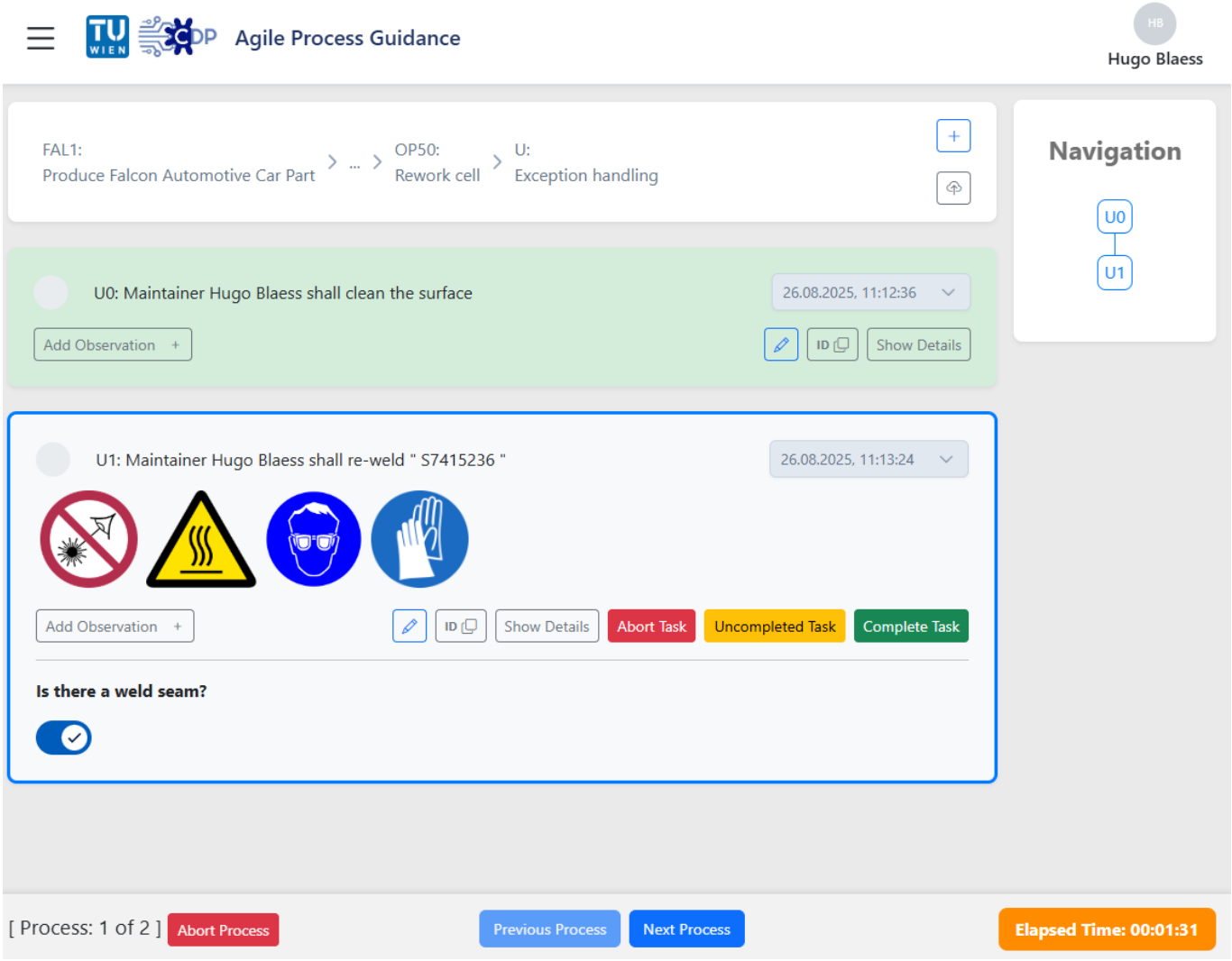
D.3: Edit Process form



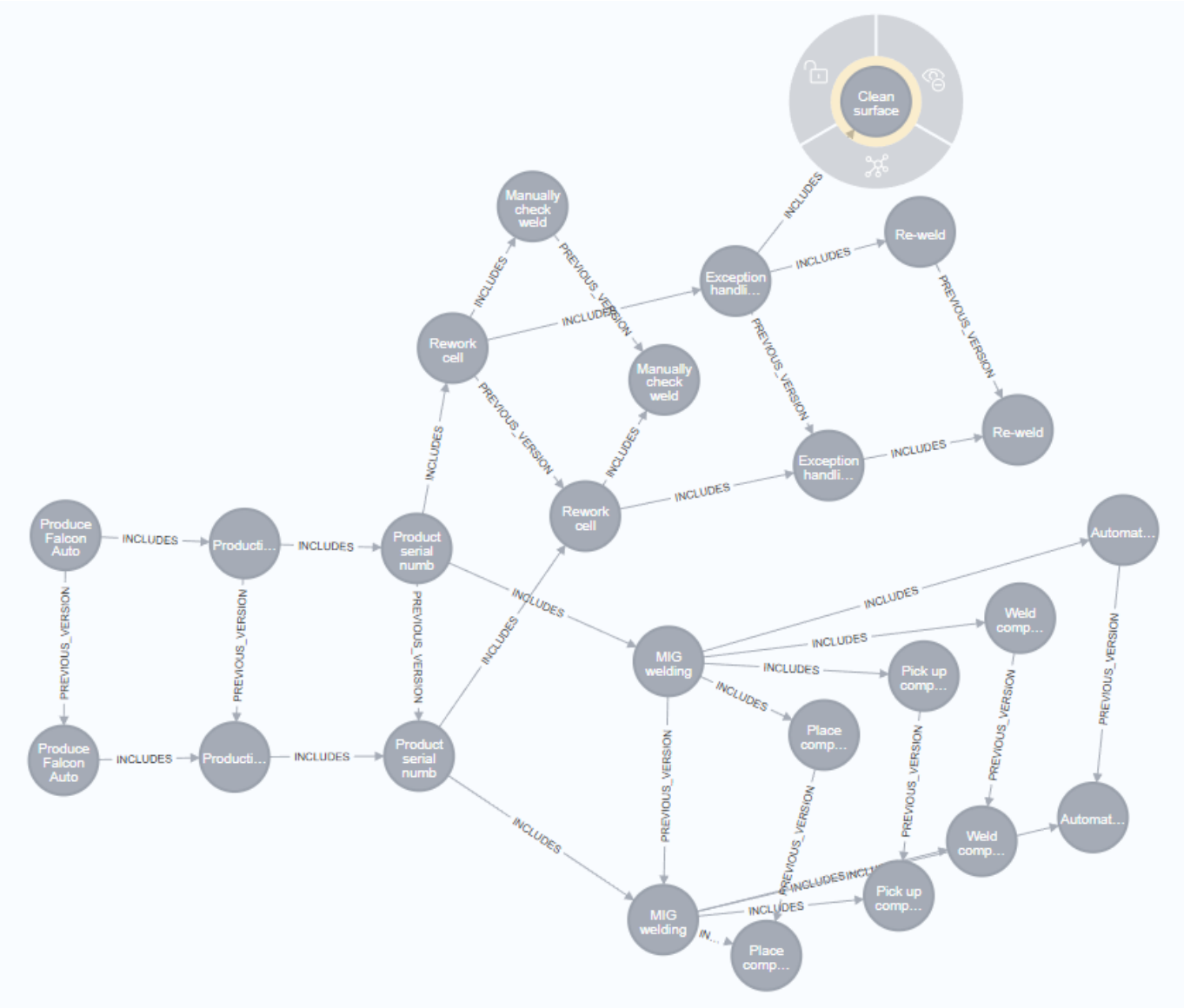
D.4: Tree view with initial and edited process



D.5: Graph view with initial and edited process



D.6: Unlocked process view



D.7: Newly added process graph

						Process Definition	
ID	Parent ID	Parent Slug (SVE	Slug	Name	Given	When	Then
d5b748fb0a8250e680a7c6a2a05f14a5			FAL1	Produce Falcon Automotive Car Part			
d1051fc316bc9f05a	d5b748fb0a825(	FAL1	W20	Production lot Week: 20, Size: 100		{{ foreach product in products }}	
7208a3cbad282540	d1051fc316bc9f	W20	S7415236	Product serial number: S7415236		Machine {{ machine }} shall produce component "{{ produ	
217063deca44b05c	7208a3cbad282	S7415236	OP20	MIG welding		Machine {{ machine }} shall prepare for MIG welding	
244d234de49880b1	217063deca44b	OP20	T20.1	Place component		Machine {{ machine }} shall place produc	MIG place {{ pro
3e09866dcae71bf1b	217063deca44b	OP20	T20.2	Weld componen	MIG place {{ product }}	Machine {{ machine }} shall perform MIG	MIG weld {{ prod
f6086ae171d4d0ca0	217063deca44b	OP20	T20.3	Automated Optic	MIG weld {{ product }}	Machine {{ machine }} shall perform opti	MIG optical insp
e0fa7caf339b87a63	217063deca44b	OP20	T20.4	Pick up compon	((MIG optical inspection {{ product }}	Machine {{ machine }} shall pick up MIG	MIG pick up {{ pr
253a81f314fdde105	7208a3cbad282	Serial Number	OP50	Rework cell	=#getCharacteristicValueForInstance	Operator {{ operator }} shall manually check property of "{{	
ea356177f3dc7378(	253a81f314fdde	OP50	T50.1	Manually check	=#getCharacteristicValueForInstance	Operator {{ operator }} shall manually che	MIG weld seam
c42eb48156d48692	7208a3cbad282	Serial Number	U	Exception handli	MIG weld seam manual check {{ prod	Maintainer {{ maintainer }} shall perform unplanned maint	
clean	7208a3cbad282	S7415236	U0	Clean surface	=#getCharacteristicValueForInstance	Maintainer {{ maintainer }} shall clean the surface	
b1c36f487df776c86	c42eb48156d48	U	U1	Re-weld		Maintainer {{ maintainer }} shall re-weld "{{ product }}"	

D.8: Export sheet

Appendix E : Interview Guide

I. Introduction & Participant Background

1. Could you please introduce yourself and describe your current role within the production environment, particularly how it relates to handling unexpected issues or "exceptions"?

2. Prior to using this new process guidance system, what were the most significant challenges you or your team faced when dealing with production exceptions? (e.g., relying on outdated manuals, needing expert intervention, slow resolution times)

II. Usability and User Experience

1. Based on your experience, how easy or difficult was it to learn and navigate this new process guidance system? Could you elaborate on what aspects were particularly intuitive or challenging?

   - Were the instructions on the interface clear?
   - Did you find the system consistent in its design and terminology?
   - Did you receive immediate feedback when performing actions?

2. The system aims to be user-friendly for all staff, including non-IT users. Do you feel the interface supports this objective? Why or why not?

III. Impact on Exception Handling and Operational Efficiency

1. How has the system influenced the way you or your team identify, diagnose, and resolve production errors in real-time? Can you provide specific examples?

   - Has it reduced the time it takes to understand what went wrong?
   - Has it made decision-making faster?

2. One of the system's objectives is to reduce the recurrence of known errors and improve the overall default rate. Do you think using the software can lead to changes in the frequency of specific production issues?

   - In what ways does the system help prevent mistakes from happening again?
   - Does it ensure best practices are consistently followed?

3. The system aims to empower less experienced staff to handle complex exceptions more confidently. In your opinion, how well does it achieve this, and how does it facilitate the transfer of expert knowledge?

   - Do you feel less experienced colleagues can operate more autonomously?

- How does the guidance support their learning “by doing”?

4. The solution centralizes and integrates various data sources, including FMEA data. How has this integration impacted your ability to access relevant information and understand the context of an exception?
   - Is it easier to connect a failure to its potential causes or corrective actions?
5. The system allows for modifications and version control of processes. How useful is this flexibility in an environment where machines, tools, and products are constantly evolving?
   - Have you needed to use the live editing feature?
   - How does this compare to previously updating static documentation?

IV. Overall Value and Future Considerations

1. Considering the solution’s overall objective to improve production line reliability and ensure productivity and quality, what is your overall assessment of its effectiveness?
2. What do you consider to be the most significant strengths of this process guidance system?
3. Are there any areas where you believe the system could be improved, or do you have suggestions for additional features that would further enhance its utility in your daily work?