TUM School of Engineering and Design
Technische Universität München

Industrial AI: A Development Methodology for Industrial Machine Learning Applications

Examined in the Context of Predictive Maintenance for Automotive Service Development

Yannic Jesko Wolf

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Die Dissertation wurde am 10.06.2024 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 07.10.2024 angenommen.

Acknowledgments

"We do these things not because they are easy,
but because we thought they were going to be easy."
– Unknown

The completion of this dissertation would not have been possible without the support of numerous individuals and institutions.
Prof. Dr. Constantinos Antoniou, my supervisor, provided the perfect balance of guidance and trust to support this work. Beyond this dissertation, he has been an invaluable source of support and inspiration for various other ideas and projects, and I look forward to the exciting opportunities the future may bring in our collaboration. I am also deeply grateful to his chair, where I was warmly welcomed and had the privilege of meeting exceptional individuals. Moreover, TU Munich provided efficient administrative support and a respectful, productive working environment.
Dr. Tetiana Aymelek, my mentor, supervisor at Porsche AG, and friend, demonstrated an exceptional ability to differentiate when I needed to be challenged and when I needed support and enablement. I hope everyone is fortunate enough to have a mentor like her during such a crucial phase of their career. The entire vehicle data analytics team fostered an outstanding working environment, from which I benefited immensely and which I thoroughly enjoyed. Porsche AG, a modern employer and technological leader, provided access to a state-of-the-art environment essential for this work's pursuit of scientific innovation.
Many students contributed to this work through their theses, delivering excellent results and advancing this work. Specifically, I would like to thank Melvin Kothe and Steven Adiputra Setiawan. The PANAMERA project team also shared their valuable experience and knowledge, significantly enriching this work. I extend my gratitude to every contributor for initiating and executing the project, which embedded this work into a broader industrial and scientific context beyond my individual scope. I am deeply grateful to Prof. Dr. Hansjörg Fromm for his excellent supervision of my master's thesis at Karlsruhe Institute of Technology, as well as for providing constructive feedback on this work and generously serving as the second examiner. Finally, Prof. Dr. Betz, with his extraordinary expertise in automotive development, led the examination of this work. I am grateful for the time and effort he invested.

So wie diese Arbeit ohne die starke und professionelle fachliche Unterstützung nicht hätte abgeschlossen werden können, wäre sie ohne die kompromisslose persönliche Unterstützung meiner Familie und Freunde ebenfalls nicht möglich gewesen.
Meine Eltern, Sibylle Wolf und Jesko Herrmann, haben jeweils auf ihre ganz eigene Weise genau das Richtige getan, um die Grundlage für mich und diese Arbeit zu schaffen. Meine gesamte Familie hat mir in dieser intensiven Zeit das volle Verständnis entgegengebracht, das ich gebraucht habe, und mich immer dort unterstützt, wo es nötig war. Dafür möchte ich von Herzen danken! Meine lieben Freunde, die Durlach Gang und die "Exilanten", haben mit ihren Ausgleichsräumen und Ablenkungen dafür gesorgt, dass ich in dieser fordernden Zeit bei Sinnen geblieben bin. Dafür danke ich euch allen von Herzen!
Üblicherweise gilt das Ende solcher Danksagungen der geduldigen und beständigen emotionalen Unterstützung durch die Lebenspartner. In diesem Fall jedoch darf ich weit darüber hinausgehen und mich für eine beeindruckende Vielfalt an Beiträgen bedanken: von der Unterstützung im Förderprojektalltag bis zur akribischsten Inhaltsprüfung, vom Alleinelassen bis zum Mitschleifen, vom Loben bis zum Erden. Als absolutes Dream Team mussten wir uns nichts mehr beweisen – aber diese Arbeit gibt uns mal wieder recht. Ohne Dich unmöglich, Yvonne Haußmann.

Von ganzem Herzen,

Yannic Jesko Wolf

Abstract

English
Predictive maintenance is currently focused on individual production machines, mainly in stationary environments. Thanks to scientific and technical advances in connected car services, it is now possible to apply predictive maintenance applications to fleet vehicles. This can improve product quality by giving feedback about vehicle component behavior in customer fleets to Research & Development departments as well as service quality by giving feedback about the actual vehicle components' conditions to customers.
The adaption of common predictive maintenance use cases to vehicle fleets needs to consider the specific characteristics of automotive development and is not thoroughly covered by research yet. The first characteristic contains the highly unequal environmental impact on the individual vehicles, caused by the multitude of different customer behavior profiles and the mobility and dynamics of the technical system. The second characteristic is the limitation in observation data by privacy regulations and broadwidth costs. A third characteristic is the fleet size. Customer vehicle fleets are large in size and therefore allow new statistical fleet analysis mechanisms.
This thesis contributes to research by developing a component repository for the creation of predictive maintenance models in automotive applications. This repository aims to generalize the similarities of the environment as far as possible while respecting its specifics as exhaustively as necessary.
This work consists of five artifacts that collectively aim to enhance the development efficiency of predictive maintenance models in automotive applications. The first artifact, stemming from a structured literature research, is a *Predictive Maintenance Framework* that validates the applicability of component-based software engineering in predictive maintenance. It demonstrates the potential value of component-based software engineering in optimizing predictive maintenance processes by identifying key requirements and benefits specific to the predictive maintenance field. The second artifact is a *Use Case Description Methodology* that generalizes the description of use cases by focusing on their inputs and outputs. This methodology facilitates the structured development of a component repository by standardizing how use cases are documented and analyzed.
The third artifact is the *Component Repository* itself, which incorporates principles of modularization, well-defined interfaces, and design principles tailored to predictive maintenance use cases. The fourth artifact is a *Development Workflow* for creating and utilizing the components within the repository. The efficiency of this workflow is validated through a discrete event simulation model and extensive sensitivity analysis, affirming its effectiveness in real-world scenarios. The fifth artifact is a *Descriptive Attributes System* that enables objects within the repository to be systematically described, compared, and refined.
Together, these artifacts create a validated component repository system that enhances development efficiency and reduces costs in a rapidly growing industrial sector. This system is particularly tailored and applicable to the automotive industry, accommodating its unique characteristics and requirements.

Deutsch
Predictive Maintenance konzentriert sich derzeit auf einzelne Produktionsmaschinen, hauptsächlich in stationären Umgebungen. Dank wissenschaftlicher und technischer Fortschritte in vernetzten Fahrzeugdiensten ist es nun möglich, Predictive-Maintenance-Anwendungen auf Fahrzeugflotten anzuwenden. Dies kann die Produktqualität verbessern, indem Feedback über das Verhalten von Fahrzeugkomponenten in Kundenflotten an die Forschungs- und Entwicklungsabteilungen gegeben wird, sowie die Servicequalität, indem Kunden Rückmeldungen über den tatsächlichen Zustand der Fahrzeugkomponenten erhalten.
Die Anpassung gängiger Predictive-Maintenance-Anwendungsfälle an Fahrzeugflotten muss die spezifischen Merkmale der automobilen Entwicklung berücksichtigen und ist in der Forschung noch nicht umfassend behandelt worden. Das erste Merkmal umfasst die stark ungleiche Umweltauswirkung auf die einzelnen Fahrzeuge, verursacht durch die Vielzahl unterschiedlicher Fahrverhaltensprofile sowie die Mobilität und Dynamik des technischen Systems. Das zweite Merkmal ist die Begrenzung der Beobachtungsdaten durch Datenschutzbestimmungen und Übertragungskosten. Ein drittes Merkmal ist die Flottengröße. Kundenfahrzeugflotten sind groß und ermöglichen daher neue statistische Flottenanalysemechanismen.
Diese Arbeit trägt zur Forschung bei, indem sie ein Komponentensystem für die Erstellung von Predictive-Maintenance-Modellen in Automobilanwendungen entwickelt. Dieses System zielt darauf ab, die Gemeinsamkeiten der Umgebung so weit wie möglich zu verallgemeinern, während ihre Besonderheiten so umfassend wie nötig berücksichtigt werden.
Diese Arbeit besteht aus fünf Artefakten, die gemeinsam darauf abzielen, die Entwicklungseffizienz von Predictive-Maintenance-Modellen in Automobilanwendungen zu verbessern. Das erste Artefakt, das aus einer strukturierten Literaturrecherche stammt, ist ein *Predictive Maintenance Framework*, das die Anwendbarkeit der komponentenbasierten Softwareentwicklung in der Predictive Maintenance validiert. Es zeigt den potenziellen Wert der

komponentenbasierten Softwareentwicklung bei der Optimierung von Predictive-Maintenance-Prozessen, indem es zentrale Anforderungen und Vorteile identifiziert, die spezifisch für das Predictive-Maintenance-Feld sind. Das zweite Artefakt ist eine *Use Case Description Methodology*, die die Beschreibung von Anwendungsfällen verallgemeinert, indem sie sich auf deren Eingaben und Ausgaben konzentriert. Diese Methodik erleichtert die strukturierte Entwicklung eines Komponenten-Repositories, indem sie standardisiert, wie Anwendungsfälle dokumentiert und analysiert werden.

Das dritte Artefakt ist das *Komponenten-Repository* selbst, das Prinzipien der Modularisierung, gut definierte Schnittstellen und Designprinzipien enthält, die auf Predictive-Maintenance-Anwendungsfälle zugeschnitten sind. Das vierte Artefakt ist ein *Entwicklungs-Workflow* zur Erstellung und Nutzung der Komponenten innerhalb des Repositories. Die Effizienz dieses Workflows wird durch ein diskretes Ereignissimulationsmodell und umfangreiche Sensitivitätsanalysen validiert, die seine Wirksamkeit in realen Szenarien bestätigen. Das fünfte Artefakt ist ein *Descriptive Attributes System*, das es ermöglicht, Objekte innerhalb des Repositories systematisch zu beschreiben, zu vergleichen und zu verfeinern.

Zusammen schaffen diese Artefakte ein validiertes Komponenten-Repository-System, das die Entwicklungseffizienz verbessert und die Kosten in einem schnell wachsenden Industriesektor senkt. Dieses System ist speziell auf die Automobilindustrie zugeschnitten und anwendbar, wobei es deren einzigartige Merkmale und Anforderungen berücksichtigt.

Keywords — Predictive Maintenance, Machine Learning, AIOps, Component-based Software Engineering, Automotive Applications, Fleet Data.

Contents

Chapter 1

Introduction

This dissertation addresses the increasingly complex field of Predictive Maintenance (PdM) in industrial applications, where the integration of advanced analytical models has become vital to operational efficiency and system reliability. Despite its widespread recognition and adoption across various sectors, the implementation of PdM often encounters significant challenges due to the diverse and dynamic nature of industrial environments. This work is motivated by the critical need to enhance the development efficiency and accuracy of PdM models, especially in contexts where system failures can result in substantial economic and safety consequences. Through a systematic exploration of methodological innovations, theoretical advancements, and practical applications, this dissertation seeks to substantiate and refine the frameworks necessary for the effective deployment of PdM strategies. The subsequent sections detail the background, motivation, specific research questions and objectives, the methodology employed, the contributions of this research, and the overall structure of the thesis.

1.1 Background

PdM represents a forward-looking approach to maintenance that anticipates equipment failures and predicts potential downtimes by harnessing the capabilities of data analytics, Machine Learning (ML), and ultimately Artificial Intelligence (AI). This strategy involves continuously monitoring equipment conditions through sensors and data collection systems, analyzing this data to detect anomalies and patterns that precede equipment failures.

Industrial AI, ML, and PdM are distinct yet interconnected terms within the realm of advanced technological applications in industrial settings. Industrial AI refers to the deployment of artificial intelligence techniques specifically tailored for industrial processes, aiming to enhance operational efficiency, productivity, and decision-making through intelligent data analysis and automation. ML, a subset of AI, involves algorithms and statistical models that enable systems to improve their performance on specific tasks through experience and data-driven learning, without explicit programming for each scenario. PdM, on the other hand, focuses on forecasting the future operational condition of machinery and systems using data analysis, ML algorithms, and domain knowledge, allowing for timely maintenance actions to prevent unexpected failures and optimize maintenance schedules.

In the automotive industry, PdM is particularly crucial due to the high costs associated with production downtimes and the complex nature of automotive manufacturing processes. Implementing PdM allows manufacturers to schedule maintenance effectively, thereby not only avoiding unplanned operational halts but also extending the lifespan of equipment and reducing maintenance-related costs. The ability to forecast potential failures is particularly advantageous in an industry driven by stringent quality standards and tight production schedules. Moreover, PdM supports the industry's agility, facilitating rapid adjustments to production processes in response to changing market demands or new product introductions (Pophaley and Vyas, 2010).

The flowchart depicted in Figure 1 illustrates the process of PdM:

1. **Start**: The process begins with the initiation of the PdM system.
2. **Data Collection**: At this stage, relevant data is collected from various sensors and monitoring systems. This data may include information about the machinery's operating conditions, performance metrics, and other relevant parameters (J. Lee, F. Wu, et al., 2014).
3. **Data Analysis**: The collected data is then analyzed using statistical methods, ML algorithms, or other analytical techniques to extract meaningful insights (Mobley, 2002a).
4. **Failure Prediction**: Based on the analysis, the system predicts potential failures or malfunctions in the machinery. This prediction enables proactive maintenance actions to be planned (Atzori, Iera, and Morabito, 2010).
5. **Decision Making**: In this stage, decisions are made regarding the necessary maintenance actions. This may include scheduling repairs, ordering replacement parts, or other actions to prevent the predicted failure (Rodič, 2017).
6. **Maintenance Action**: The planned maintenance actions are carried out. This may involve repairing or replacing components, adjusting operating parameters, or other actions to ensure the continued efficient operation of the machinery (X. Li et al., 2021).

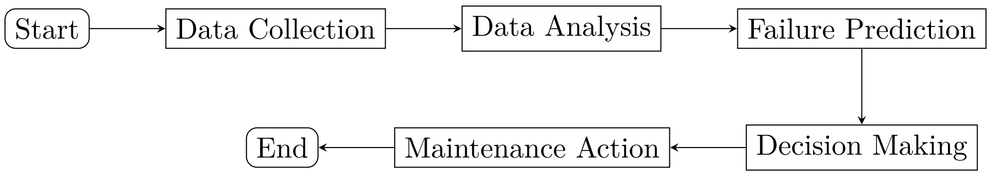


Figure 1: Flowchart of the Predictive Maintenance Process (own figure).

7. **End**: The process concludes once the maintenance actions are successfully completed.

The flowchart represents a high-level overview of the PdM process, highlighting the key stages and the flow of information and actions from one stage to the next. It emphasizes the data-driven nature of PdM and the integration of advanced analytical techniques to support proactive maintenance actions. This approach aligns with the broader trends in Industry 4.0 (I4.0) and the growing importance of data and analytics in modern manufacturing and automotive applications (Poor, Basl, and Zenisek, 2019).

PdM plays a key role in sustainable manufacturing and production systems (Achouch et al., 2022), and it is considered the leading one in maintenance technology (C. Chen et al., 2021). The concept of PdM is also extended to I4.0, representing the next evolution step in industrial maintenance development (Poor, Basl, and Zenisek, 2019).

In the initial stages of technological advancement, the design and development of PdM models were decentralized within development teams of individual components. Each team, working in isolation, created models that were often highly similar to one another, if not identical. This approach is reflective of the early maintenance strategies, where Condition-based Monitoring (CbM) and PdM were developed to provide valuable information for establishing maintenance policies (Zio and Compare, 2013). The decentralization led to a focus on short-term strategies and economic factors, often favoring corrective maintenance over PdM (Ruiz and Guevara, 2020).

The emergence of Over-The-Air (OTA) data transmission technology marked a significant shift in this paradigm. With an increase in customer data availability through OTA, a trend for centralization of PdM model development began to take shape (Dalzochio et al., 2020). This centralization allowed the various aspects of PdM, which were previously handled by individual teams, to be managed by a specialized central team (Mi et al., 2021).

The move towards centralization brought with it several benefits, primarily by leveraging economy of scale factors (Yiwei Wang et al., 2017). By consolidating PdM under one umbrella, organizations could ensure consistency, foster innovation, and rationalize efforts (Achouch et al., 2022). However, this centralization also introduced new challenges, particularly the need for a higher level of organization and management (Mi et al., 2021).

The centralization of PdM model development represents not just a technological advancement but also a strategic shift in how organizations approach maintenance (Poor, Basl, and Zenisek, 2019). It emphasizes the growing importance of data-driven decision-making and highlights the need for agile, adaptable structures that can respond to rapidly changing technological landscapes (Bousdekis, Apostolou, and Mentzas, 2020).

The shift from decentralization to centralization in PdM model development symbolizes the industry's move towards a more unified, efficient, and innovative approach (Achouch et al., 2022). By bringing together different aspects of PdM under one roof, organizations can achieve greater synergies and capitalize on the potential efficiencies of scale (Yiwei Wang et al., 2017). However, this shift also calls for a new way of thinking about organizational structures, methodologies, and technologies, all of which are central to the focus of this research (Dalzochio et al., 2020).

PdM, as a concept, has two distinct applications: as a manufacturing application and as a Cyber-Physical Product Service System (CPPSS).

In the context of manufacturing, PdM is utilized to monitor and analyze the condition of machinery and equipment within a production environment. It leverages data analytics, ML, and other advanced technologies to predict potential failures and schedule timely maintenance (J. Lee, F. Wu, et al., 2014). This approach enhances efficiency, reduces downtime, and supports the overall optimization of the manufacturing process (Mobley, 2002a).

On the other hand, as a CPPSS, PdM extends beyond the manufacturing floor to customer vehicles as products of the automotive industry.

A CPPSS in the automotive industry represents a sophisticated integration of Cyber-Physical System (CPS) and Product Service Systems Product-Service System (PSS), extending traditional manufacturing boundaries to include real-time monitoring, PdM, and personalized services directly within customer vehicles. This system not only connects with the vehicle's onboard systems to collect data on performance metrics but also analyzes this data to predict and pre-empt potential issues, thereby enabling proactive maintenance strategies that enhance vehicle performance and customer experience. Such integration allows CPPSS to offer highly responsive and personalized maintenance services, meeting the evolving demands of modern automotive consumers and improving overall operational efficiency (Rodič, 2017; X. Li et al., 2021; Atzori, Iera, and Morabito, 2010; Porter and Heppelmann, 2015).

These two applications of PdM, though interconnected, represent different facets of the technology, each with its unique challenges and opportunities. The manufacturing application focuses on optimizing industrial processes, while the CPPSS approach emphasizes enhancing product functionality and customer engagement.

PdM presents both unique challenges and opportunities in the context of manufacturing and as a CPPSS.

Challenges

- **Technological Hurdles**: Implementing PdM requires advanced technologies such as data analytics, ML, and real-time monitoring. Integrating these technologies can be complex and costly (J. Lee, F. Wu, et al., 2014).
- **Data Privacy and Ethics**: The collection and analysis of data in PdM raise concerns about privacy and ethical considerations, especially in the context of customer vehicles (Atzori, Iera, and Morabito, 2010).
- **Market Trends**: Adapting to rapidly changing market demands and technological advancements requires continuous innovation and flexibility in PdM strategies (Poor, Basl, and Zenisek, 2019).

Opportunities

- **Enhanced Efficiency**: PdM enables organizations to optimize maintenance schedules, reduce downtime, and enhance overall efficiency (Mobley, 2002a).
- **Customer-Centric Services**: As a CPPSS, PdM allows for personalized and responsive maintenance services for automotive consumers, enhancing customer experience (Porter and Heppelmann, 2015).
- **Strategic Innovation**: The integration of PdM with I4.0 offers opportunities for strategic innovation and alignment with future industrial trends (Rodič, 2017).

The challenges and opportunities associated with PdM reflect the multifaceted nature of this approach. While technological and ethical challenges must be addressed, the potential benefits in efficiency, customer engagement, and strategic alignment present significant opportunities for organizations in the manufacturing and automotive sectors.

The improvement of the Technology Readiness Level (TRL) in the development of PdM models for automotive fleet applications holds significant relevance and impact. The TRL is a systematic measurement framework used to assess the maturity and readiness of a particular technology. It consists of a scale ranging from 1 to 9, where each level represents a different stage of technological development, from the initial concept (TRL 1) to full commercial deployment (TRL 9). The TRL serves as a valuable tool for researchers, developers, and stakeholders to understand the progress, risks, and challenges associated with a technology, guiding decision-making and resource allocation (Boardman and B. Sauser, 2006; Ramirez-Marquez and B. J. Sauser, 2009).

Enhancing the TRL of PdM models leads to more refined and efficient development processes, translating to cost savings by eliminating unnecessary redundancies and allocating resources more effectively (J. Lee, F. Wu, et al., 2014). This improvement also enables faster implementation and deployment, supporting rapid response to market demands and aligning with the fast-paced nature of the automotive industry (Mobley, 2002a).

Furthermore, fostering a collaborative environment allows practitioners to exchange knowledge and insights, leading to continuous learning and innovation in PdM model development (Poor, Basl, and Zenisek, 2019). The insights gained from automotive fleet applications are not confined to this domain alone; they can be translated and applied to other industrial AI application areas, broadening the scope and impact of PdM models (Rodič, 2017). Improving modularization and reusability in the development of PdM models is closely tied to the advancement of the TRL.

This work contributes to the advancement of TRL in several key ways. By integrating a novel framework that emphasizes agile development, modular design, and continuous integration, this work enhances the speed and efficiency of developing PdM models. The proposed framework breaks down the development process into discrete, manageable components that can be worked on simultaneously by distributed teams, thereby accelerating the overall TRL progression. Additionally, the incorporation of a simulation-based validation strategy ensures iterative testing and refinement of components before their integration into the larger PdM system. This systematic approach not only increases the maturity of individual components but also facilitates a smoother transition through the TRL stages, ultimately achieving higher levels of technology readiness in a more structured and reliable manner. The collaborative environment fostered by this framework allows for continuous knowledge exchange and innovation, further driving the advancement of PdM models across various industrial AI applications.

Modularization is the practice of dividing a system into smaller, self-contained modules. This approach allows for parallel development, testing, and maintenance of individual components. Improving modularization in PdM model development enhances the TRL by enabling more efficient validation, reducing complexity, and fostering innovation (Baldwin and Clark, 2000), (Parnas, 1972).

Reusability refers to the ability to use existing components or modules across different contexts or applications. Enhancing reusability in PdM model development promotes the TRL by leveraging proven solutions, minimizing redundancy, and accelerating the development process (Cohen and Krut, 2018), (Biggerstaff and Perlis, 1989), (Frakes and Terry, 1996).

The relationship between improving modularization, reusability, and the TRL can be understood through the following aspects:

- **Cost Efficiency**: By improving modularization and reusability, organizations can reduce development costs and resource allocation, thereby enhancing the TRL (Lim, 1994).
- **Speed to Market**: Enhanced modularization and reusability enable faster development cycles, reducing time-to-market and aligning with the rapid pace of technological advancement (Clements and Northrop, 2002).
- **Quality and Reliability**: Improved modularization and reusability contribute to higher quality and reliability in PdM models, as validated components can be reused with confidence (Mili, 2002).
- **Adaptability**: The flexibility afforded by modularization and reusability allows for easier adaptation to changing requirements, supporting the continuous evolution of the TRL (Szyperski, Gruntz, and Murer, 2009).

In conclusion, the improvement of modularization and reusability in PdM model development is a strategic approach to enhance the TRL. It fosters a more agile, cost-effective, and robust development process, positioning PdM as a key technology in the automotive industry and beyond (Ravichandar, 2022).

The relevance and impact of improving the TRL in PdM model development extend beyond mere technological advancement. It signifies a strategic shift towards more cost-effective, agile, and collaborative practices. The potential for cross-domain applicability further underscores the importance of this advancement in shaping the future landscape of industrial AI and PdM.

1.2 Motivation

PdM represents a significant advancement in industrial operations, offering the potential to predict and prevent system failures before they occur. The economic stakes associated with unplanned downtime in critical industries—such as manufacturing, energy, and transportation—underscore the importance of reliable and efficient maintenance strategies.

Despite its potential, the implementation of PdM systems often faces significant challenges. These include complexities in data collection and processing, variability in industrial settings, and the need for models that can adapt dynamically to changing conditions. Additionally, there remains a substantial gap between theoretical PdM models and their practical application in real-world settings. Advancements in PdM methodologies can lead to substantial benefits for industries, ranging from increased operational efficiency and cost savings to enhanced safety and environmental sustainability. For instance, better predictive capabilities can reduce the frequency of equipment checks and maintenance, thus minimizing energy use and prolonging equipment life. This dissertation specifically addresses gaps in methodological development, the integration of robust data analysis frameworks, and the application of these models in diverse industrial environments. There is a particular need for a framework that not only predicts system failures but also integrates seamlessly with existing industrial processes. The choice of this research topic stems from a deep-seated interest in enhancing industrial efficiency and reliability, which are vital to global economic stability and safety. This work is also driven by a commitment to bridging the gap between theoretical research and practical, tangible outcomes.

By addressing these critical challenges and gaps, this research aims to significantly enhance the PdM landscape, providing actionable insights and tools that can be adopted across various industries to mitigate risks and optimize performance.

1.3 Research Questions and Objectives

The formulation of precise Research Questions (RQs) and objectives is vital in guiding the trajectory of any academic inquiry (Creswell, 2014). In the context of this study, which focuses on PdM applications, the research is structured around three primary RQs. These RQs serve as the foundation upon which the research methodology, data collection, and analysis are built (Yin, 2018).

The methodology for designing these RQs involves a systematic approach that includes the following steps (Kitchenham, 2004):

1. **Literature Review**: A comprehensive review of existing literature to identify gaps and opportunities for research (Webster and Watson, 2002).
2. **Consultation with Experts**: Discussions with experts in the field to gain insights not readily available in existing literature.
3. **Preliminary Data Collection**: Exploratory studies and surveys conducted to further

refine the research focus (Easterbrook et al., 2008).

4. **Alignment with Objectives**: Ensuring that the RQs align well with the research objectives and scope (Wohlin, 2012).
5. **Feasibility Assessment**: Evaluation of the resources required to answer the RQs (Runeson and Höst, 2009).
6. **Clarity and Specificity**: Finalization of RQs that are clear, specific, and researchable (Stol, Ralph, and Fitzgerald, 2016).

This research aims to address the following three RQs:

1. Is it possible to classify the development of a PdM application in a framework?
2. Is it possible to design a component repository using the similarities while respecting the individual aspects of PdM?
3. Is it possible to reduce development costs for PdM applications?

Each of these RQs is designed to tackle specific challenges in the field of PdM, and collectively, they aim to provide a comprehensive solution to the complexities involved in PdM applications. The RQs are structured to provide a precise and validated approach to the study, as depicted in Figure 2.

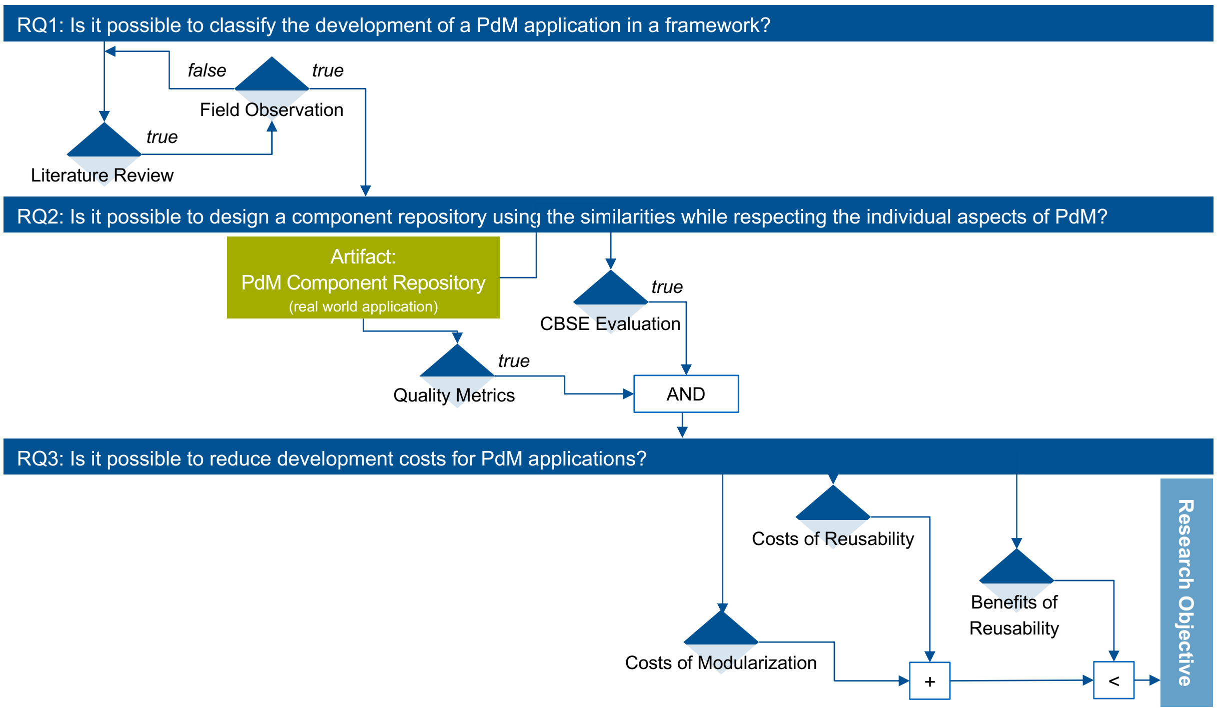

RQ1: Is it possible to classify the development of a PdM application in a framework?
false
true
Field Observation
true
Literature Review
RQ2: Is it possible to design a component repository using the similarities while respecting the individual aspects of PdM?
Artifact:
PdM Component Repository
(real world application)
true
CBSE Evaluation
true
Quality Metrics
AND
RQ3: Is it possible to reduce development costs for PdM applications?
Costs of Reusability
Benefits of Reusability
Costs of Modularization
+
<
Research Objective

Figure 2: Research question structure and process flow (own figure).

RQ1: The initial RQ begins with a literature review aimed at uncovering various approaches for categorizing the development of PdM applications. If a viable approach is found, field observations are conducted to validate its applicability and, if necessary, refine it based on real-world conditions.

RQ2: The focus of the second RQ is on establishing a repository for componentized PdM models. The repository undergoes two key evaluations:

1. **CBSE Evaluation**: This assesses whether the PdM model comply with Component-Based Software Engineering (CBSE) standards for componentization.
2. **Quality Metrics**: This evaluation measures the performance, reliability, and overall quality of the PdM model, ensuring that the modularization process does not compromise their predictive accuracy.

Positive outcomes in both evaluations are required for the validation of this RQ.

RQ3: The third RQ explores the cost-benefit trade-offs involved in modularizing and reusing PdM applications. The investigation encompasses:

1. Costs associated with Modularization
2. Expenses related to Reusability
3. Long-term Benefits of Reusability

The overarching research objective focuses on ensuring that the benefits of reusability surpass the combined costs of modularization and reusability. This objective serves as the guiding principle for the design and evaluation of the artifacts, as well as for the economic assessments conducted in the research.

In summary, the research is structured around three central RQs, each designed to address distinct yet interrelated facets of PdM. The process flow for each RQ is precisely planned to ensure both academic rigidity and practical relevance, as outlined in Figure 2. The first RQ seeks to categorize PdM application development through literature review and field observation. The second RQ aims to establish a repository of modularized PdM models, ensuring they meet both CBSE standards and quality metrics. The third RQ looks into the economic aspects of modularization and reusability in PdM applications. Collectively, these RQs provide a comprehensive framework for advancing the field of PdM through methodical investigation and validation.

1.4 Research Methodology

In light of the complex and interdisciplinary nature of developing a component repository process for PdM, this research employs Design Science Research (DSR). The choice of DSR is motivated by several factors. First, the methodology is well-suited for the development and validation of innovative artifacts, aligning with the research's aim to create a component repository process (Hevner et al., 2004). Second, DSR accommodates the involvement of multiple stakeholders, including practitioners in an Original Equipment Manufacturer (OEM) and academic partners in a publicly funded research consortium, thereby ensuring that the research outcomes are both academically accurate and practically relevant (Peffers et al., 2007). Third, the interdisciplinary nature of this work, which integrates expertise from PdM and data science, is well-supported by DSR's flexibility in incorporating theories and methods from various domains (Venable, Pries-Heje, and Baskerville, 2012). Lastly, the iterative approach of DSR aligns with the research's continuous refinement process, involving frequent iteration loops of design and evaluation in a real-world environment (Vaishnavi, Vaishnavi, and Kuechler, 2015). Therefore, DSR serves as an apt methodology for this research, providing a structured yet flexible framework for addressing its multifaceted research questions and objectives.

In the context of this research, DSR is particularly instrumental in the development and validation of the component repository process for PdM. The methodology's focus on artifact creation consistently aligns with the research's objective to break down PdM models into modular and component-based structures. By employing DSR, the research is able to systematically design, implement, and evaluate the component repository, ensuring that it meets the specific requirements and constraints imposed by the real-world automotive OEM environment (Hevner et al., 2004).

The DSR methodology offers a multitude of advantages in the field of information systems and technology research. One of its most notable features is its structured framework, which includes well-defined stages such as problem identification, artifact design, and evaluation (Hevner et al., 2004). This structured approach ensures research rigidity, providing a systematic pathway for tackling complex problems and enhancing the validity and reliability of research outcomes. Another advantage is its focus on artifact creation, which can encompass models, frameworks, software, and hardware solutions (Peffers et al., 2007). These artifacts serve a dual purpose: they contribute to academic knowledge while also offering practical solutions to real-world problems, effectively bridging the gap between academia and industry. Furthermore, DSR is inherently iterative, promoting cycles of design, evaluation, and refinement (Vaishnavi, Vaishnavi, and Kuechler, 2015). This iterative nature allows for the continuous improvement of the artifact, making it adaptable to emerging technologies and changing requirements. Additionally, DSR encourages the use of multiple evaluation methods, such as case studies, experiments, and simulations, offering a more holistic understanding of the artifact's utility and effectiveness (Gregor and A. R. Hevner, 2013). Lastly, DSR is highly interdisciplinary, allowing for the integration of theories and methods from various domains, thereby enriching both the research process and its outcomes (Venable, Pries-Heje, and Baskerville, 2012). Overall, DSR provides a comprehensive, flexible, and stringent methodology for conducting research that is both academically sound and practically relevant.

This research follows the key stages of DSR as outlined by Hevner et al. (2004). The process begins with the identification of the research problem, followed by the definition of the objectives for a solution. The next stage involves the design and development of the artifact, in this case, the component repository process for PdM. This is followed by a demonstration phase where the artifact is tested in a real-world automotive OEM setting. Subsequent evaluation is conducted through iterative loops involving both researchers and practitioners, ensuring the artifact's utility and effectiveness. Finally, the research findings and contributions are communicated to the relevant academic and industrial communities.

1.5 Contribution

This work introduces a refined methodology for the creation of PdM models, which is particularly tailored to technical environments, emphasizing efficiency through the application of component-based software principles and a strong focus on reusability. While the methodology is broadly applicable, it offers specific benefits in technically constrained domains such as automotive product development, effectively utilizing the limited spectrum of available input data types and the specific PdM objectives (J. Lee, Bagheri, and Kao, 2015).

The contributions of this work are detailed as follows:

1. A structured literature review covering both general and automotive-specific PdM, which substantiates the proposed PdM framework and supports the hypothesis of limited and recurring functional building blocks for PdM models.
2. An elaborate methodology to define use cases for PdM models within the automotive industry, which thoroughly details every technical and knowledge dependency, structures interfaces, and establishes standards for managing PdM model development.
3. The development of a component repository structure as a collation of components. This repository leverages the PdM framework portrayed in the literature review and the methodological description of use cases, providing detailed guidelines necessary for the planning

and execution of PdM model development. It includes a comprehensive set of functional modules that describe each phase of the model development process, complete with defined facets and interfaces between modules.

4. The introduction of an innovative approach for component development through a priori reusability analysis, which applies reusability measurement metrics as core design principles and validates the approach using real-world use cases.
5. The design and validation of a workflow model that supports the development of PdM models by utilizing the structured component repository and the novel component development methodology. This model is substantiated through stringent Discrete Event Simulation (DES), providing quantitative evidence of the contributions' validity and the efficiency benefits, thereby forming the basis of the cost function for this research.
6. The formulation of a formal descriptive attributes system for the input and output attributes of components developed within the workflow model. This system not only enhances usability but also sets the stage for future enhancements in development efficiency.

This work culminates in a sophisticated methodology that enhances the efficiency and usefulness of PdM model development in the automotive sector, integrating comprehensive framework validation via detailed analysis and simulations, and establishing a robust architectural foundation for use case development and component repository structuring. This comprehensive approach is proven to quantitatively demonstrate efficiency gains and support scalability and adaptability in PdM practices.

1.6 Outline of the Thesis

This section provides a structured outline of the thesis, detailing the sequential arrangement and the interconnections between the various chapters that collectively encapsulate the research objectives, methodologies, and findings of this work.

The *Literature Review* chapter 2 lays the foundational knowledge and situates this research within the existing body of scientific work, structured to methodically unfold the scope and depth of this thesis. Initially, section 2.1 introduces specific research questions and objectives focused on exploring the integration of CBSE principles in the development of PdM models, aiming to identify potential efficiency gains. This is followed by section 2.3, where fundamental concepts and terminologies pertinent to AI, PdM, and CBSE are portrayed to ensure clarity for all readers. In section 2.4, the thesis presents an exhaustive overview of existing AI development methodologies and frameworks, highlighting their alignment or divergence from the practices proposed in this work. Next, section 2.5 examines the principles of CBSE, emphasizing the prerequisites needed for efficiency gains through its application. The chapter culminates in section 2.6 with the presentation of the first artifact of this work—a PdM Framework developed through structured literature research using the Natural Language Processing (NLP) topic modeling technique Latent Dirichlet Allocation (LDA), illustrating the practical application of the theoretical concepts discussed.

The *Methods* chapter 3 explains the diverse methodologies employed in this thesis, beginning with an introduction to the chosen research design, DSR, which is thoroughly detailed in section 3.1. Following this foundational discussion, section 3.2 describes the research environment, setting the stage for understanding the context in which the study is conducted. The artifacts developed during the research are then extensively analyzed and classified in section 3.3, providing a deep dive into the tools and techniques crafted as part of this thesis. Finally, section 3.4 discusses the ethical considerations inherent to this research, ensuring that all aspects of the study adhere to stringent ethical standards.

The *Execution* chapter 4 details the practical application of the methodologies and frameworks developed in this work, structured into several key sections. It begins with section 4.1, which presents the Use Case Description Methodology, outlining how it captures and specifies the requirements for PdM and AI development. This is followed by section 4.2, where the Component Repository is discussed, explaining its role in enhancing reusability and standardization of components. Section 4.3 conducts the a priori analysis that underpins the repository's effectiveness, assessing component compatibility and integration before implementation. In section 4.4, the Component Creation Workflow is introduced, describing the procedural steps for developing and integrating new components. Finally, section 4.5 examines the Descriptive Attributes System, which categorizes and describes the properties of each component, facilitating easier navigation and utilization within the repository.

The *Discussion and Outlook* chapter 5 offers a comprehensive evaluation and forward-looking perspective on the research presented in this thesis. Initially, the discussion in section 5.1 looks into the implications of the findings, providing a critical analysis of how the results contribute to the broader field of study. Subsequent to this analysis, section 5.2 acknowledges the limitations encountered throughout the research, offering a transparent examination of potential biases, constraints, and areas where further investigation is needed. The chapter concludes with section

5.3, where future research directions are proposed, outlining prospective studies that could build upon the foundational work established here and explore new avenues within the domain of PdM and CBSE.

Chapter 2

Related Literature

The Literature Review section of this thesis undertakes a systematic exploration of the multifaceted domain of PdM, with a specific focus on assessing the applicability and relevance of the central artifact of this work: the development of a component repository based on CBSE principles. This comprehensive review begins with introducing the core *Research Questions and Objectives*, investigating the feasibility and existing efforts of applying CBSE principles in PdM model development in subsection 2.1. Following this, an outline of the methodology adopted for conducting structured literature research is presented, detailing the data sources and selection criteria in subsection 2.2 *Overview and Methodology.* To ensure clarity and uniform understanding, the review first explains key *Definitions and Terminologies* relevant to PdM, ML, and CBSE in subsection 2.3. It then progresses to examine *Theoretical and ML Frameworks*, highlighting the current state of research and identifying gaps, particularly in the application of these frameworks in industrial PdM in subsection 2.4. The exploration of CBSE principles and their origin provides insights into their potential applicability in PdM in subsection *Component-Based Software Engineering* 2.5. *Empirical Studies* are then presented to evaluate if PdM fulfills the requirements for a CBSE approach. This culminates in the creation of an innovative framework, detailed in subsection 2.6, which represents the first-ever approach to structuring the field of PdM using a quantitative methodology based on related research. This framework systematically organizes PdM into four distinct phases, each encompassing two to three specific facets, thereby offering a comprehensive and nuanced understanding of the field. The section culminates with a *Conclusion and Discussion* in subsection 2.7, synthesizing the findings, addressing the research questions, and critically analyzing the strengths and weaknesses of the methodologies employed.

2.1 Research Questions and Objectives

The formulation of structured research questions is a fundamental step in scholarly inquiry, serving as a compass that guides the entire research process. According to Booth et al. (2016), well-defined research questions are essential for outlining the scope of a study, providing clarity and direction. Furthermore, Levy and J. Ellis (2011) emphasizes that research questions should be both specific and feasible, ensuring that the investigation remains focused and achievable. In the context of this literature review, the research questions are crafted to specifically address the intersection of CBSE and PdM, a domain where existing literature is either sparse or unexplored. The precision and relevance of these questions are not only vital in guiding the literature review but also crucial in identifying gaps in current knowledge and potential areas for innovation.

Central to this literature research are two thoroughly formulated research questions that guide the exploration into the synergy between CBSE and PdM. The first question, "*What are the existing endeavors, if any, that have sought to enhance PdM development by integrating CBSE principles?*", is designed to systematically uncover and analyze previous attempts and methodologies in this domain. This inquiry is key for understanding the extent and effectiveness of CBSE applications in PdM development. In the event of limited or non-existent prior efforts, the investigation naturally progresses to the second question: "*Considering the theoretical frameworks and principles of CBSE, how feasible is its application in the development of PdM systems?*", which aims to assess the theoretical viability and potential strategies for incorporating CBSE in PdM. This question is instrumental in identifying theoretical foundations and potential innovative approaches for the application of CBSE in PdM. The objectives derived from these questions aim to provide a thorough literature review and a critical analysis of the theoretical possibilities for applying CBSE in PdM, thereby offering valuable insights and contributions to the field.

2.2 Overview and Methodology

This research embarks on an extensive literature review to thoroughly understand the research landscape surrounding PdM.

The literature research for the subsections on Definitions and Terminologies (2.3), Theoretical & ML Frameworks (2.4), and CBSE (2.5) is conducted using a methodology adapted from Webster and Watson (Webster and Watson, 2002). This approach involves constructing a search query using a collection of key terms, which is then used to narrow down relevant scientific publications in various databases. Following Webster and Watson's methodology, both forward and backward searches are conducted to add additional articles (Webster and Watson, 2002). Due to the vast amount of literature available, the focus is on the quality of research results and comprehensive coverage of areas relevant to this work, rather than completeness. A concept matrix is then used to comparatively display a selection of the reviewed articles (Webster and Watson, 2002).

In the Empirical Studies section (2.6), a thoroughly structured literature analysis is undertaken. This analysis is vital not only for acquiring a comprehensive understanding of the PdM domain but also for addressing the critical literature research question: Can the principles of CBSE be effectively applied to the field of PdM? This inquiry forms the cornerstone of the research, guiding the systematic examination of existing literature. The structured approach ensures a thorough and methodical exploration of the PdM landscape, examining various studies to discern whether the methodologies and principles inherent in CBSE have been, or could be, successfully integrated into PdM practices. This analysis is instrumental in identifying potential avenues for the application of CBSE in PdM, thereby contributing significantly to the development of a novel framework aimed at enhancing the efficiency and effectiveness of PdM models. The primary data source is the SCOPUS[1] database, a comprehensive repository of scientific papers across diverse disciplines. Abstracts from approximately 700 scientific papers are thoroughly mined, specifically those containing the keyword `Predictive_Maintenance` and categorized under the `Engineering` label in SCOPUS.

In addition to thematic analysis, a descriptive analysis of the literature data, including authors and journals, is conducted. This analysis is essential for understanding the research dynamics, such as identifying key contributors, influential journals, and the evolution of the field over time (Webster and Watson, 2002).

The critical step following data collection is to analyze the thematic structures within this substantial corpus. To achieve this, a topic modeling approach via LDA is utilized. LDA facilitates the identification and categorization of primary topics and themes within the abstracts, providing structured and insightful perspectives into prevailing research trends and focal points in PdM within the engineering sphere.

A novel, multi-level LDA analysis is then employed to further dissect and structure the extensive literature field related to PdM. The initial phase of this analysis, the first-layer LDA, segments the corpus into six distinct topics, establishing a broad thematic categorization. Subsequently,

[1]SCOPUS Database. `https://www.scopus.com/`. Accessed on [11/08/2023].

a second-layer LDA analysis is thoroughly applied to each of these six topics. This layered approach, innovative in its application and not extensively documented in existing literature, draws upon the flexibility and scalability of LDA as a topic modeling tool (Blei, Ng, and Michael I. Jordan, 2003). By sequentially applying LDA at multiple levels, the methodology is designed to reveal more granular subtopics within each broader category, thus offering a deeper, more detailed understanding of the literature. This comparative analysis across topics and subtopics is instrumental in identifying thematic patterns, relationships between topics, and potential research gaps. Such a structured, hierarchical analysis is especially beneficial in complex fields like PdM, where it can effectively uncover and describe the underlying thematic structures in the literature.

2.3 Definitions and Terminologies

Within the realm of PdM research, several terms and concepts stand out as particularly salient. To ensure clarity and a shared understanding throughout this literature review, some of the key terms are defined below:

Predictive Maintenance: Refers to a proactive maintenance strategy that utilizes data analysis, ML, and predictive analytics to forecast when equipment failures might occur, thereby enabling timely maintenance actions to prevent unplanned downtime (Mobley, 2002b). This strategy significantly contrasts with Preventive Maintenance (PvM), which schedules maintenance activities at regular, predetermined intervals based solely on statistical life expectancy rather than the actual condition of the equipment. PdM leverages real-time data collected from sensors and historical maintenance records to tailor maintenance tasks specifically to the individual conditions and operational behaviors of each piece of equipment. This targeted approach helps in optimizing maintenance resources and costs by performing maintenance only when needed, rather than on a fixed schedule. Unlike Reactive Maintenance (RM), which addresses equipment failures post-occurrence, PdM aims to preemptively identify and mitigate potential failures, thus enhancing the overall reliability and availability of equipment and reducing operational risks (Gulati, 2012; Swanson, 2001). Furthermore, PdM supports a strategic alignment with business objectives by improving asset utilization and extending the life of equipment, which are critical for achieving operational excellence in maintenance-intensive industries (Hashemian and Bean, 2011).

Component-Based Software Engineering: CBSE - also refered to as Component-Based

Development - is defined as an approach to software development that emphasizes the design and assembly of pre-existing software components into a larger software system, thereby easing reuse and modularization. This software engineering paradigm seeks to improve software quality and reduce development time by leveraging reusable components that have been tested and verified independently (Heineman and Councill, 2001; Crnkovic and Larsson, 2003). According to Heineman and Councill (2001), CBSE focuses on the assembly of component infrastructures, middleware, and application-level components that meet well-defined interface specifications. This approach is characterized by its emphasis on the separation of concerns in software design, which allows for the independent development and testing of software components (Szyperski, Gruntz, and Murer, 2009). Furthermore, Crnkovic and Larsson (2003) suggest that the utilization of CBSE can lead to more efficient and manageable software systems, as components can be developed and maintained in parallel by different teams. The primary advantage of CBSE lies in its potential to significantly reduce the complexity of software development while improving scalability and maintainability, thus making it an ideal approach for large-scale and complex software systems.

Code Component Reusability: Refers to the practice of designing software components in such a way that they can be reused in different systems or applications without or with minimal modifications (Y. Kim and Stohr, 1998). Reusability is a key principle in CBSE, where the development process focuses on creating modular, self-contained components that encapsulate specific functionality and can be easily integrated into multiple software projects. This approach not only speeds up the software development process by reducing the amount of code that needs to be written from scratch but also enhances software quality and maintainability by reusing well-tested components (Sametinger, 1997). Effective reusability requires that components have well-defined interfaces, are sufficiently general to be used in varied contexts, and are documented to allow their integration and use by other developers. CBSE frameworks promote reusability by providing standards and tools that help in the cataloging, retrieval, and integration of reusable components, thus fostering an ecosystem where developers can share and leverage each other's work to build complex systems more efficiently (Szyperski, Gruntz, and Murer, 2009). Moreover, reusability is closely related to other software quality attributes such as modularity, encapsulation, and maintainability, making it a vital aspect of contemporary software engineering practices (Bosch, 2000).

These definitions serve as foundational knowledge, ensuring a clear and consistent understanding while further exploring the elaborate landscapes of PdM research.

2.4 Theoretical & ML Frameworks

ML frameworks are indispensable in today's technologically-driven landscape, streamlining complex computational processes and fostering the growth of intelligent systems across various sectors, including PdM. The increasing importance of these frameworks correlates with the maturing TRL of ML. As the technology readiness of ML advances, there's an escalating demand for standardization to harness the technology's full potential (M. I. Jordan and T. M. Mitchell, 2015).
Comparable fields such as mathematical and statistical science, as well as computer and information engineering, have witnessed similar maturation trajectories. These disciplines, in their respective growth phases, have accentuated the need for frameworks to boost efficiency. Essentially, ML is tracing a parallel path, reinforcing the significance of structured platforms to navigate the complexity of its applications.
Moreover, the congruence in ML use cases across diverse application fields underscores the vast potentials of these frameworks. The universality of certain ML tasks, irrespective of the domain, suggests that standardized processes can catalyze efficiency, minimize redundancies, and pave the way for more rationalized and robust solutions (Goodfellow, Bengio, and Courville, 2016; Bishop, 2006).
The concept of Machine Learning Operations (MLOps) has gained significant traction both in academia and industry in recent years (Makinen et al., 2021; Symeonidis et al., 2022; Pölöskei, 2021).
In academia, a growing number of research papers focus on the challenges and solutions associated with MLOps, such as model versioning, deployment, and monitoring (Symeonidis et al., 2022; Renggli et al., 2021; Mboweni, Masombuka, and Dongmo, 2022).
Concurrently, industry also shows a keen interest in adopting MLOps practices to restructure ML workflows, improve model performance, and ensure robustness and scalability (Makinen et al., 2021; Pölöskei, 2021; Mallela et al., 2023).
The rising trend of MLOps in academic literature can be attributed to the increasing complexity of ML models and the need for robust, scalable solutions for deploying these models into production (Symeonidis et al., 2022; Renggli et al., 2021). Moreover, the integration of MLOps with PdM applications is becoming a focal point of research, as it promises to enhance the efficiency and

reliability of PdM systems (Makinen et al., 2021; Mallela et al., 2023).

In industry, companies are increasingly investing in MLOps platforms and tools to manage the end-to-end ML lifecycle. This investment is driven by the need to accelerate the deployment of ML models and to ensure that the models are maintainable, interpretable, and robust (Makinen et al., 2021; Pölöskei, 2021; Mallela et al., 2023).

Before delving into MLOps, it's essential to understand its precursor, DevOps. Originating from software development, DevOps promotes a cultural shift, emphasizing the collaboration between software developers and IT operations (Makinen et al., 2021). It aims to shorten the system development lifecycle, provide continuous delivery, and achieve high software quality (Makinen et al., 2021). With the increasing complexity and dynamism of ML projects, there arose a need for a specialized subset of DevOps, leading to the birth of MLOps (Makinen et al., 2021).

MLOps, an evolution of DevOps tailored for ML, serves as a foundational pillar in bridging the gap between ML development and operations. This practice promotes continuous integration, delivery, and training of ML models, thereby fostering iterative development while ensuring optimal operational performance. Key components of MLOps include:

- Continuous Integration (CI): Integrating code changes into a shared repository, ensuring code quality, and enabling rapid iterations (Zhou, Yu, and B. Ding, 2020).
- Continuous Delivery (CD): Seamless transition of ML models from development to production, enhancing deployment pace (Zhou, Yu, and B. Ding, 2020).
- Continuous Training (CT): Periodic retraining of ML models to tackle data drifts and model obsolescence (Banerjee et al., 2023).

By adhering to these principles, MLOps ensures model reproducibility, traceability, and automation. This is fundamentally important, especially for domains like PdM, where timely and accurate predictions are fundamental (Makinen et al., 2021; Mallela et al., 2023).

The adoption of ML frameworks in PdM is an advancing area of research and application. One of the notable advancements is the use of digital twin frameworks that integrate ML and physics-based modeling for PdM (Kunzer, Berges, and Dubrawski, 2022). This approach offers a comprehensive view of system health and performance, thereby enhancing maintenance strategies. Another significant contribution is a framework that maps sensors and variables related to equipment load cycles, such as turbines in hydroelectric power plants, to pave the way for PdM (Vallim Filho et al., 2022). This methodology transforms the problem of forecasting future maintenance needs into a binary classification problem.

Automated ML processes are also gaining traction in PdM. For instance, state-based transfer learning and ensemble methods are employed to automate the PdM process, reducing user interaction time and decision-making complexity (Larocque-Villiers, Dumond, and Knox, 2021). Furthermore, Deep Reinforcement Learning (DRL)-based frameworks promise effective resource management for Industrial Internet of Things (IIoT) applications, demonstrating superiority in terms of convergence efficiency and simulation performance (Ong et al., 2022).

These advancements indicate the growing success of ML frameworks in PdM, but they also highlight the need for a more integrated and comprehensive approach to fully realize the potential benefits.

Integrating ML frameworks with PdM applications presents a set of unique challenges that can be broadly categorized into technical and organizational aspects. On the technical side, issues such as scalability, data integration, and model training are prevalent. Scalability is crucial as PdM applications often require real-time analysis of large datasets. Data integration poses challenges in harmonizing data from diverse sources, including sensors and historical maintenance records. Model training involves selecting appropriate algorithms and tuning parameters for accurate failure predictions (Dalzochio et al., 2020; Vollert, Atzmueller, and Theissler, 2021).

Organizational challenges include resistance to adoption and change management. Employees may be hesitant to trust ML-based systems, and there may be a lack of expertise in managing these advanced technologies. Moreover, there is often a gap between current ML practices and the specific requirements of PdM, such as the need for interpretable models and real-time decision-making (Herrmann, 2020; Theissler et al., 2021).

In the context of the literature review, the work by Theissler et al. (2021) serves as a vital reference, especially given its focus on the automotive industry, which aligns closely with the industrial applications discussed in this thesis. The paper investigates the challenges and use-cases of implementing ML-enabled PdM in the automotive sector. These insights are particularly relevant for addressing the first research question concerning the classification of PdM applications within a framework. Moreover, the paper's discussion on the challenges in PdM could offer valuable perspectives for the second research question on designing a component repository. While the paper does not explicitly discuss cost aspects, its focus on practical use-cases provides an implicit understanding of the economic implications, thereby contributing to the third research question on cost reduction in PdM applications (Theissler et al., 2021).

The integration of ML into PdM is an emerging trend that holds significant promise for the future of industrial applications. Component-based engineering, a methodology that promotes the reuse of software components, is gaining traction in the realm of ML and PdM (Khorsheed and Beyca, 2021; Donghwan Kim, S. Lee, and Daeyoung Kim, 2021). The potential for a holistic PdM solution lies in the amalgamation of ML frameworks with component-based engineering. This approach not only enhances the performance of equipment but also reduces maintenance costs (Shamayleh, Awad, and Farhat, 2020; Kiangala and Z. Wang, 2020). However, it is important to note that while ML methods such as deep learning are being explored, traditional feature engineering-based approaches show to be more effective in some cases (Silvestrin, Hoogendoorn, and Koole, 2019). Future research should focus on overcoming the challenges related to data availability and feedback collection, as well as improving the accuracy and reliability of predictive models (Dalzochio et al., 2020; Bouabdallaoui et al., 2021). The ultimate goal is to develop a comprehensive PdM framework that can be universally applied across various industrial sectors. In the realm of PdM, the integration of ML frameworks shows promising results in various industrial applications, including automotive, manufacturing, and energy sectors (Khorsheed and Beyca, 2021; Kiangala and Z. Wang, 2020; Donghwan Kim, S. Lee, and Daeyoung Kim, 2021). These frameworks have been effective in detecting failures before they occur, reducing downtime, and optimizing maintenance interventions (Khorsheed and Beyca, 2021; Kiangala and Z. Wang, 2020). However, there is a notable absence of a unified approach for the integration of ML in PdM, which presents challenges such as data availability, unstructured maintenance logs, and the need for cost-sensitive learning (Silvestrin, Hoogendoorn, and Koole, 2019; Cadavid et al., 2022; Spiegel et al., 2018).

This research aims to fill this gap by proposing a PdM framework that not only leverages ML algorithms for fault prediction but also incorporates a component repository to remodel the development process. The proposed framework addresses the individual aspects of PdM applications while exploiting their similarities, thereby offering a pathway to reduce development costs (Dalzochio et al., 2020; Lwakatare et al., 2020). As PdM continues to evolve in the context of I4.0, the need for a unified, efficient, and cost-effective approach becomes increasingly critical (Dalzochio et al., 2020; Abidi, Mohammed, and Alkhalefah, 2022).

2.5 Component-Based Software Engineering

CBSE is a paradigm in software engineering that emphasizes the design and construction of software systems through reusable and interchangeable components. This approach aims to improve the efficiency, reliability, and maintainability of software development processes. In the context of this thesis, CBSE holds particular relevance for developing efficient PdM frameworks in industrial applications, especially in the automotive sector. This subsection will explore the principles, historical development, and applications of CBSE, highlighting its significance in PdM and AI.

CBSE has evolved significantly over the years. Initially, software development was largely monolithic, where applications were built as single, indivisible units. The advent of CBSE marked a paradigm shift, emphasizing modularity and reusability. This evolution was guided by the eight laws of software evolution, which provide a phenomenological description of the evolutionary behavior observed in various software systems (Lehman and Ramil, 2000). Over time, CBSE has incorporated various formal methods and techniques, adapting to the needs of safety-critical systems and complex software architectures, including those in industrial and automotive applications (Lehman and Ramil, 2000; Williams, 1991).

The core principles of CBSE serve as the foundation for its methodology. These principles include reusability, modifiability, and maintainability. Reusability allows for the same component to be used in multiple applications, thereby reducing development time and costs. Modifiability enables easy adaptation and extension of components, promoting system evolution. Maintainability focuses on the ease with which a system can be updated or repaired. These principles collectively aim to improve the overall efficiency and reliability of software systems, including those in PdM and AI (Crnkovic, Sentilles, et al., 2011; Atkinson and Hummel, 2012).

The economic rationale behind CBSE is deeply rooted in the concept of economic efficiency, achieved through the use of reusable components. By employing a CBSE approach, organizations can significantly reduce development time and costs, thereby achieving higher economic efficiency. This is especially relevant in the context of PSS, where modular and reusable components enable rapid adaptation to changing customer needs or technological advancements. The CBSE approach allows for more flexible and cost-effective PSS, providing a competitive advantage in fast-paced markets, including the automotive sector (Stallinger et al., 2002; Emmerich and Kaveh, 2002; Berkovich et al., 2014).

The key elements in CBSE include components, interfaces, and component frameworks. Components are the modular building blocks that encapsulate specific functionalities. Interfaces define the interaction points between components, specifying what a component can do and how it can be used. Component frameworks provide the architectural backbone that supports the integration and interaction of components. These elements collectively enable the development of robust and scalable software systems (Pour, 1998; Y. Wu, Dai Pan, and Mei-Hwa Chen, 2001).

Componentization methods in CBSE are diverse, ranging from architectural patterns to specific algorithms. One common approach is to use design patterns like the Factory Method or the Singleton Pattern to encapsulate component creation logic. Additionally, algorithms for efficient retrieval of component repositories help component selection and integration (Bawa and I. Kaur, 2016; Sadrani et al., 2023). Moreover, metrics like 'reusability-ratio' and 'interaction-ratio' are employed to estimate component reliability and execution time (Tiwari, S. Kumar, and Matta, 2020). These methods and metrics not only update the componentization process but also contribute to the system's overall reliability and efficiency, particularly in PdM and AI.

Evaluating the effectiveness of a CBSE approach involves various metrics and methods. Metrics such as 'component cohesion' and 'component coupling' are commonly used to assess the quality of individual components. Additionally, methodologies like Preference Ranking Organization METHod for Enrichment Evaluations (PROMETHEE) provide a quantitative framework for component selection and trust-building (K. Kaur and Singh, 2014). These evaluation techniques offer insights into the reliability and efficiency of a CBSE system and are particularly relevant for PdM and AI.

The PROMETHEE methodology employs a multi-criteria decision analysis to rank components based on various attributes like reliability, efficiency, and cost. It quantifies the qualitative selection and evaluation of software components, thereby providing a more nuanced understanding of component quality. The methodology uses pairwise comparisons to evaluate the trade-offs between different components, which helps in making more informed decisions. This method not only builds trust in the selected components but also offers a robust framework for component selection in CBSE. The methodology comprises several steps:

Step 1: Identification of Criteria - The first step involves identifying the criteria against which the components will be evaluated. These criteria can include attributes like reliability, efficiency, and cost.

Step 2: Scoring - Each component is scored based on these criteria. The scoring can be done using various scales, such as a Likert scale or a numerical range.

Step 3: Pairwise Comparisons - The methodology employs pairwise comparisons to evaluate the trade-offs between different components. Each component is compared with every other component based on each criterion.

Step 4: Preference Functions - Preference functions are used to quantify the preference of one component over another for each criterion. These functions can be linear, exponential, or Gaussian, among others.

Step 5: Aggregation - The preferences are then aggregated to produce a global preference score for each component. This is usually done using weighted averages of the criteria.

Step 6: Ranking and Selection - Finally, the components are ranked based on their global preference scores, and the best-suited components are selected for integration into the system.

This comprehensive approach not only builds trust in the selected components but also offers a robust and quantifiable framework for component selection in CBSE. The PROMETHEE methodology thus provides a nuanced understanding of component quality, making it an invaluable tool for practitioners and researchers alike (K. Kaur and Singh, 2014).

The increasing intersection of CBSE with PdM and AI is a focal point of contemporary interdisciplinary research (Hasterok and Stompe, 2022). This section illustrates the transformative impact of CBSE principles in architecting PdM and AI systems that are scalable, efficient, and adaptable.

CBSE finds a compelling application in PdM for rotating machines, where neural models are integral for real-time monitoring, fault detection, and diagnostics (El Mahdi et al., 2022). The architecture employs CBSE to compartmentalize the neural model, thereby enabling modular updates and adaptability (Xia Cai et al., 2000).

The architecture presented in the paper is a quintessential example of CBSE application in PdM for rotating machines (El Mahdi et al., 2022). It modularizes the neural model into distinct, function-specific components such as data acquisition, fault detection, and diagnostics.

This modularization strategy consistently aligns with CBSE principles, offering enhanced flexibility and scalability. Each component can be independently updated or replaced, ensuring that the system remains agile and up-to-date (El Mahdi et al., 2022).

The architecture thoroughly adheres to CBSE metrics of component cohesion and coupling. High cohesion within components and low coupling between them are strategically maintained to ensure system robustness and maintainability (El Mahdi et al., 2022).

The application of CBSE principles in this architecture paves the way for scalable and efficient PdM solutions. The modular architecture enables seamless integration into broader PdM systems, thereby offering a viable strategy for cost and time reduction in development (El Mahdi et al., 2022).

In hydraulic systems, CBSE is employed in a digital twin framework, which synergizes virtual models with real-world components to significantly enhance diagnostic accuracy over traditional, non-interactive simulation models (L. Wang et al., 2022; Wijayasiriwardhane, Lai, and K. Kang, 2011).

In the realm of AI, neural models are increasingly being modularized using CBSE principles. This modularization facilitates easier updates and consistent integration of new algorithms and techniques (Hasterok and Stompe, 2022).

The Process Model for AI Systems Engineering (PAISE) model stands as a seminal contribution to AI systems engineering (Hasterok and Stompe, 2022). It advocates for a component-wise development strategy for complex systems, thereby enabling parallel development processes and accelerating the development cycle.

The PAISE model introduces interdisciplinary checkpoints, which serve as validation junctures to precisely test component Dependencies (Dp). This ensures alignment with overarching system requirements and leads to a more refined component specification (Hasterok and Stompe, 2022).

The PAISE model accentuates the importance of parallelizing domain-specific development processes, a feature that is invaluable in multi-disciplinary projects. This parallelization optimizes resource utilization and accelerates the system's time-to-market (Hasterok and Stompe, 2022).

The PAISE model's component-wise approach and emphasis on parallelization make it particularly relevant for PdM applications. By modularizing the predictive algorithms and integrating them as components within a larger system, the PAISE model offers a scalable and efficient approach to PdM (Hasterok and Stompe, 2022).

The concept of digital twins is also being integrated into AI systems, providing a virtual representation that can be used for various simulations and analyses (Conmy and Bate, 2014).

While the application of CBSE in PdM and AI is promising, it is not devoid of challenges. Key issues include component compatibility, data security, and the need for real-time processing (Grunske, Kaiser, and Reussner, 2005). Nonetheless, the modular architecture inherent to CBSE provides avenues for incremental updates and improvements. These modular capabilities are especially pertinent in the context of PdM and AI, offering a pathway for overcoming these challenges (Loiret et al., 2011).

The integration of CBSE into PdM and AI is an emergent field with substantial promise for the development of systems that are not only scalable but also flexible and efficient (Cai, Lyu, and Wong, 2002). As advancements continue, it is anticipated that increasingly sophisticated components and frameworks will emerge, thereby amplifying the capabilities and reach of PdM and AI systems (Urtado, H. Y. Zhang, and Vauttier, 2010).

In conclusion, CBSE serves as a robust and versatile framework for the development of PdM and AI systems. Its modular architecture lends itself well to complex, multi-disciplinary projects, including those in the automotive sector. However, a notable research gap persists. While there is successful application of CBSE in general AI and specific PdM scenarios, there is limited research at the intersection of CBSE and PdM in industrial settings, particularly in the automotive industry. This gap is significant as automotive applications often involve unique challenges such as real-time requirements, high reliability demands, and elaborate system interactions, all of which could benefit from a CBSE approach. Bridging this gap has the potential to not only advance the PdM field but also to showcase the adaptability and effectiveness of CBSE across diverse application domains, including automotive. As the field matures, the fusion of CBSE with PdM and AI is poised to produce increasingly sophisticated and efficient systems.

2.6 Execution and Evaluation of a structured Literature Review

In order to conduct a comprehensive literature review on PdM, a systematic approach to gather relevant publications is employed. This involved querying academic databases using specific search strings, applying inclusion and exclusion criteria.

The data for this research is obtained through an automated crawling process targeting the Scopus database. This approach is not only efficient but also widely recognized in the scientific community for building high-quality collections and indices of scientific papers (Hoff and Mundhenk, 2001). Automated crawling of databases like Scopus is validated in various studies. For instance, Chapman, Morgan, and Gartlehner (2010) confirmed the validity of a semi-automated search method using the Scopus database, which identified the same studies as traditional approaches. Furthermore, tools like *pybliometrics* ease reproducibility and enhance data integrity for researchers using Scopus data (Rose and Kitchin, 2019).

The use of Scopus as a sole data source for citation-based research is also endorsed, especially when citations in conference proceedings are sought (Meho and Rogers, 2008). Moreover, the merging of Scopus with other databases like Web of Science shows to provide more comprehensive bibliometric analysis (Echchakoui, 2020).

The search string used for the crawling process is detailed in 2.1, and the approach aligns with the current best practices in the field.

Lst. 2.1: Display of the search string applied for the database query.

```
TITLE–ABS–KEY("predictive maintenance") AND (SUBJAREA(ENGI))
```

The data obtained from the Scopus database consists of *723* results, spanning the years 2007 to 2023. The resulting *pandas* DataFrame (DF) includes the following columns: `title`, `year`, `authors`, `abstract`, `publication_name`, `aggregation_type`, `DOI`, and `Scopus_ID`. The distribution of publications per year is depicted in Figure 3, showing a growing trend in the literature related to PdM. It is important to note a limitation in the data for the year 2023, as the collection occurred during that year, possibly resulting in incomplete information.
Despite the oldest publications in the field being much older[2], the available data reaches far enough into the past to cover all current trends. This aligns with the findings of (Mongeon and Paul-Hus, 2016), which emphasize the importance of capturing a comprehensive time span to accurately reflect the evolution and current state of a research field.
The increase in publications over time reflects the growing interest and advancements in PdM, a trend that is observed in various engineering and industrial domains (J. Lee, Bagheri, and Kao, 2015). The data's temporal coverage ensures a robust understanding of both historical developments and contemporary innovations in the field (Waltman, van Eck, and van Raan, 2012).

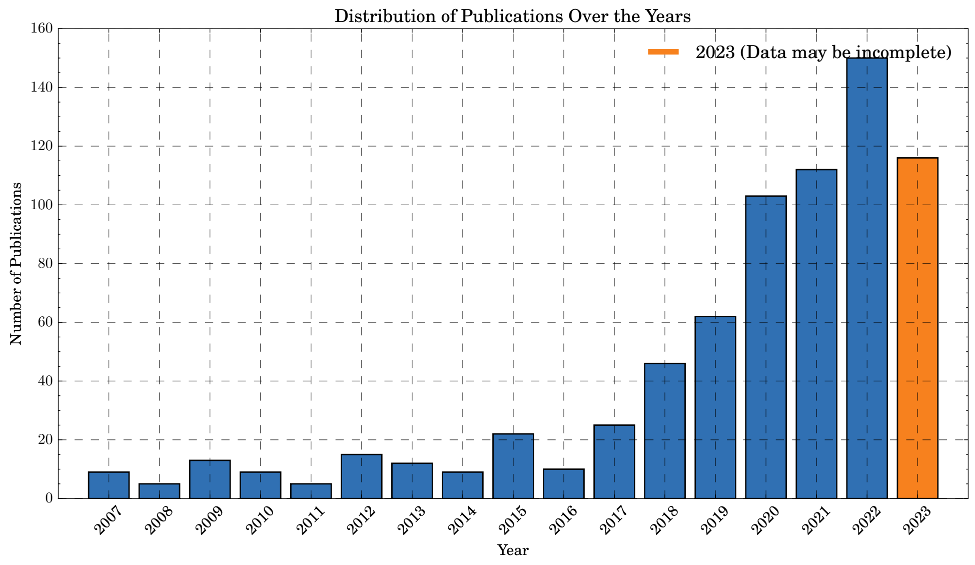


Figure 3: Publications per year, showing the search results of the query `TITLE-ABS-KEY("predictive maintenance") AND (SUBJAREA(ENGI))` in the *Scopus* database (own figure).

Table 1 presents the top 10 journals in which the publications related to the research are found. These journals include well-known titles such as "Reliability Engineering & System

[2]E.g., Mobley (2002a)

Safety", "Procedia Manufacturing", and "Mechanical Systems and Signal Processing". The counts represent the number of articles retrieved from each journal during the structured literature research. It is noteworthy that the research spans a total of *104* different journals, reflecting the interdisciplinary nature of the field and the wide range of publications that contribute to the study of PdM and related areas.

Publication Name	**Count**
Reliability Engineering & System Safety	68
Procedia Manufacturing	35
Mechanical Systems and Signal Processing	27
Measurement	27
Computers & Industrial Engineering	26
Expert Systems with Applications	23
Engineering Applications of Artificial Intelligence	23
Computers in Industry	23
Journal of Manufacturing Systems	19
Engineering Failure Analysis	12

Table 1: The 10 Journals in which most of the publications that were found with the query `TITLE-ABS--KEY("predictive maintenance" ) AND (SUBJAREA(ENGI))` in the *Scopus* database appeared, with their respective count.

Table 2 presents the top 5 authors in the dataset, along with their respective institutions, and the number of their publications. The number of first author publications per author is almost equal, with each of the top four authors having 4 publications and the fifth author having 3.
The geographical distribution of these authors spans across various countries, including the United States, Australia, Greece, and Norway. These locations can be considered as expert hubs for PdM research. For instance, the integration of PdM technologies with optimal maintenance scheduling models is effective in real-world scenarios, including in the United States (Yildirim, Sun, and N. Z. Gebraeel, 2016). Australia shows advancements in PdM using Artificial Neural Networks (ANNs) (T. Wang et al., 2008). Greece contributes to the development of PdM frameworks that can be applied directly in the industrial field (Donghwan Kim, S. Lee, and Daeyoung Kim, 2021). Norway is involved in the development of methodologies for PdM based on sensor data (Naskos et al., 2019).
The total number of unique authors in the dataset is *2529*, and the total number of unique first authors is *657*. This distribution indicates a wide range of contributors to the field, but the number of authors with more than one publication is quite limited. This could be attributed to the specialized nature of PdM research and the diverse applications of PdM across various industries, such as manufacturing, energy, transportation, and more (J. Lee, Lapira, et al., 2013; Z. Zhao, Koutsopoulos, and J. Zhao, 2018).

Rank	Author	Institution	Number of Publications
1	Ethan Wescoat[a]	International Center for Automotive Research, Clemson University, United States	4
1	Ke Feng[b]	School of Mechanical and Mechatronic Engineering, University of Technology Sydney, Australia	4
1	Panagiotis Aivaliotis[c]	Laboratory for Manufacturing Systems & Automation, University of Patras, Greece	4
1	Oluseun Omotola Aremu[d]	School of Mechanical and Mining Engineering, The University of Queensland, Australia	4
5	Haidar Hosamo Hosamo[e]	University of Agder, Norway	3

Table 2: The 5 authors with the most publications that were found with the query `TITLE-ABS--KEY("predictive maintenance" ) AND (SUBJAREA(ENGI))` in the *Scopus* database, with their respective institutions and the number of their publications.

[a](Wescoat, Bangale, et al., 2023; Wescoat, Krugh, Jansari, et al., 2023; Wescoat, Mears, et al., 2020; Wescoat, Krugh, Henderson, et al., 2019)

[b](Feng, Ji, Ni, et al., 2023; Feng, Ji, Y. Zhang, et al., 2023; Feng, Ni, et al., 2022; Feng, Ji, Y. Li, et al., 2022)

[c](Aivaliotis, Arkouli, Georgoulias, et al., 2021; Aivaliotis, Arkouli, Kaliakatsos-Georgopoulos, et al., 2021; Aivaliotis, Xanthakis, and Sardelis, 2020; Aivaliotis, Georgoulias, et al., 2019)

[d](Aremu, Cody, et al., 2020; Aremu, Hyland-Wood, and McAree, 2020; Aremu, Hyland-Wood, and McAree, 2019; Aremu, Palau, et al., 2018)

[e](Hosamo, Nielsen, et al., 2023b; Hosamo, Nielsen, et al., 2023a; Hosamo, Svennevig, et al., 2022)

2.6.1 Authorship Trends in Predictive Maintenance

The field of PdM witnesses a significant increase in collaborative research, as evidenced by the rising number of authors in scientific publications. This trend contrasts with the authorship patterns observed in related software-engineering publications, highlighting the unique interdisciplinary nature of PdM.

In PdM, the complexity of integrating various technologies such as data analytics, ML, and real-time monitoring demands collaboration across different faculties and expertise. A study titled "The rising trend in authorship" by Aboukhalil (2014) predicts that by 2034, publications will boast an average of 8 authors.

On the other hand, the field of software engineering also experiences an increase in the number of authors, but at a different rate. A study titled "Authorship trends in software engineering" by J. M. Fernandes (2014) provides evidence that the number of authors in software engineering articles is increasing on average around +0.40 authors per decade (J. M. Fernandes, 2014). Another study titled "Evolution in the number of authors of computer science publications" by J. M. Fernandes and Monteiro (2017) confirms that all computer science areas, including software engineering, witness an increase in the average number of authors.

The higher average number of authors in PdM publications indicates a greater need for interfaculty exchange and networking. This collaboration fosters innovation, enhances the quality of research, and reflects the multifaceted nature of PdM, which requires expertise in various domains such as engineering, data science, and AI. In conclusion, the authorship trends in PdM and software engineering reveal the distinct collaborative dynamics in these fields as displayed in figure 4.

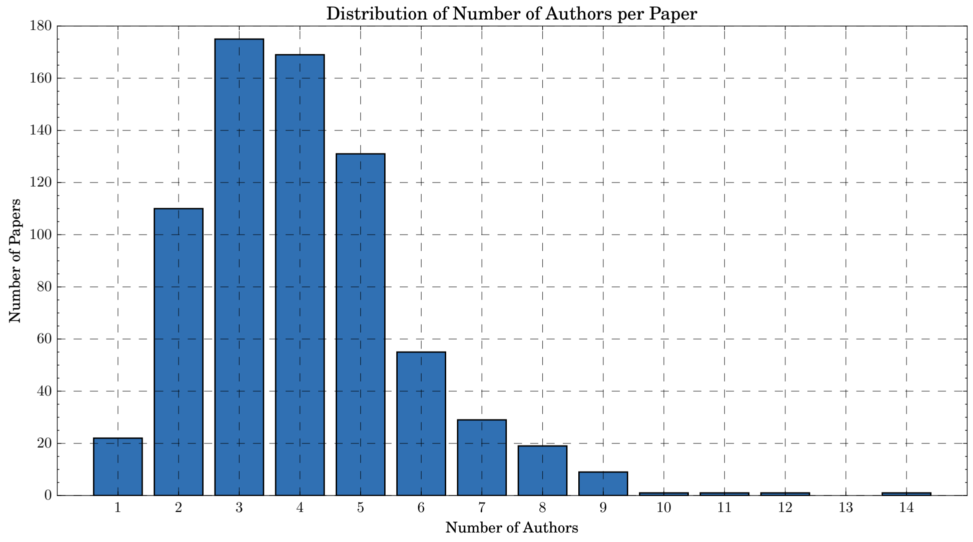


Figure 4: Distribution of the number of authors per paper for the publications that were found with the query `TITLE-ABS-KEY("predictive maintenance") AND (SUBJAREA(ENGI))` in the *Scopus* database (own figure).

The analysis of author networks in the field of PdM revealed several key insights, including the identification of leading researchers, prevalent collaboration patterns, and the interdisciplinary nature of the field. This information can guide future research collaborations and provide a roadmap for navigating the complex landscape of PdM research.

2.6.2 Topic Modeling Methodology

In this work, topic modeling is applied to structure the extensive dataset described earlier. The aim is to automatically identify and categorize the underlying themes or topics within the corpus of scientific abstracts. This approach is particularly useful for making sense of large volumes of unstructured data and is successfully applied in various domains, including the analysis of scientific abstracts (Gerlach, Peixoto, and Altmann, 2018).

Topic modeling is a technique in text mining that aims to automatically identify topics present in a text corpus. It is widely used in various domains such as marketing, security, education, and management (W. Ding, Ishwar, and Saligrama, 2014). One of the most popular algorithms for topic modeling is LDA (Wallach et al., 2009).

LDA is a probabilistic model that assumes each document is a mixture of a small number of topics and that each word in the document is attributable to one of the document's topics. The model is robust under various conditions and integrates both subjective and objective knowledge

(Blei, Ng, and Michael I. Jordan, 2003).

The mathematical formulation of LDA is as follows:

$$p(\theta, z, w \mid \alpha, \beta) = p(\theta \mid \alpha) \prod_{n=1}^{N} p(z_n \mid \theta) p(w_n \mid z_n, \beta) \tag{2.6.1}$$

Where:

- θ is the topic distribution for document d.
- z is the topic for the n^{th} word in document d.
- w is the specific word.
- α and β are hyperparameters.
- N is the total number of words in document d.

The LDA model is based on the following generative process for each document d in a corpus D:

- Choose $N \sim$ Poisson(ξ).
- Choose $\theta \sim$ Dirichlet(α).
- For each of the N words w:
  - Choose a topic $z \sim$ Multinomial(θ).
  - Choose a word w from $p(w|z, \beta)$, a multinomial probability conditioned on the topic z.

The LDA algorithm aims to discover the topic structure that optimizes the likelihood of the observed corpus, often through iterative methods (Wallach et al., 2009). The model's parameters can be estimated using various techniques, including Gibbs sampling and variational inference (Arora et al., 2012).

In this work, coherence scoring is employed to determine the optimal number of topics for the LDA model. Coherence scoring is a metric that quantifies the quality of topics generated by a topic model. A higher coherence score indicates a more coherent and interpretable topic (Morstatter and H. Liu, 2016). It is applied in various contexts, such as e-commerce recommendations, machine translation (Xiong and M. Zhang, 2013), and social media analysis (Blair, Bi, and Mulvenna, 2020).

The coherence score is calculated based on the semantic similarity between high-ranking words within each topic. It shows to be effective in discriminating the usefulness of topics (Contreras-Piña and Ríos, 2016) and is considered a reliable measure for evaluating the quality of topics in the absence of ground truth data (Doogan and Buntine, 2021). The formula for calculating topic coherence is often based on word co-occurrence and pair-wise word similarities (Omar et al., 2015).

$$\text{Coherence Score} = \sum_{i,j \in W, i \neq j} \text{similarity}(w_i, w_j) \tag{2.6.2}$$

Where W is the set of high-ranking words in a topic, and similarity(w_i, w_j) is a measure of the semantic similarity between words w_i and w_j.

The choice of coherence scoring in this work is motivated by its proven effectiveness in various applications and its ability to provide a more interpretable and coherent set of topics.

2.6.3 Overview of Topic Coherence Measures

In the context of LDA, a coherence score serves as a quantitative metric to evaluate the quality of topics generated by the model (Blei, Ng, and Michael I. Jordan, 2003; Röder, Both, and Hinneburg, 2015). The necessity for such a score arises from the challenge of determining how well the topics make sense, both semantically and contextually ("Automatic Evaluation of Topic Coherence" 2006). Traditional evaluation methods often involve manual inspection, which is not only time-consuming but also subjective (Chang et al., 2009). Coherence scores aim to provide an automated, objective measure of topic quality (Röder, Both, and Hinneburg, 2015). Generally, a coherence score works by examining the top words in each topic and measuring how frequently these words co-occur in a given corpus (Mimno et al., 2011). Various coherence measures such as UMass Coherence Index (UCI), UMass Coherence (UMASS), Normalized Pointwise Mutual Information (NPMI) and Confirmatory V-measure (C_V) employ different mathematical formulations and statistical properties to capture this co-occurrence information (Röder, Both, and Hinneburg, 2015). When choosing a coherence score, it is essential to consider the specific requirements of the research question at hand, the characteristics of the data, and the computational resources available. Comparative studies often recommend using multiple coherence scores to cross-validate the robustness of the topics generated by the LDA model (O'Callaghan et al., 2015).

The paper "Exploring the Space of Topic Coherence Measures" by Röder, Both, and Hinneburg (2015) provides insights into different coherence scores used in topic modeling. The coherence scores UCI, UMASS, NPMI and C_V are explained as follows:

The UCI coherence, calculated using Pointwise Mutual Information (PMI), measures the average log probability that pairs of top words co-occur more frequently than would be expected by chance. A higher UCI score generally indicates that the top words within the topic are more coherent and semantically related (Röder, Both, and Hinneburg, 2015, p. 2). It is based on word co-occurrence counts derived from documents constructed by a sliding window that moves over an external reference corpus like Wikipedia.[3]

$$C_{UCI} = \frac{1}{N \cdot (N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \mathrm{PMI}(w_i, w_j) \tag{2.6.3}$$

$$\mathrm{PMI}(w_i, w_j) = \log \left(\frac{P(w_i, w_j) + \epsilon}{P(w_i) \cdot P(w_j)} \right) \tag{2.6.4}$$

where N is the number of top words for a topic and ϵ is a small constant to avoid the logarithm of zero (Röder, Both, and Hinneburg, 2015, p. 2).

UMASS coherence focuses on the conditional probability of co-occurrence of top word pairs. Unlike UCI, it considers the order of words and is generally easier to compute. A higher UMASS score suggests that the preceding words provide a good context for the succeeding words within the topic (Röder, Both, and Hinneburg, 2015, p. 2). UMASS coherence uses an asymmetrical confirmation measure between top word pairs, which accounts for the ordering among the top words of a topic.

$$C_{UMass} = \frac{1}{N} \sum_{i=1}^{N} \log \left(P(w_i | w_j) \right) \tag{2.6.5}$$

where N is the number of top words for a topic (Röder, Both, and Hinneburg, 2015, p. 2).

NPMI coherence is an extension of PMI and is normalized to fall within the range of -1 to 1. It provides a balanced measure that accounts for the frequency and rarity of word co-occurrence, making it a more robust metric for assessing topic quality (Röder, Both, and Hinneburg, 2015, p. 3). NPMI is an improvement over PMI and performs better than UCI coherence.

$$C_{NPMI} = \frac{1}{N \cdot (N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \mathrm{NPMI}(w_i, w_j) \tag{2.6.6}$$

where N is the number of top words for a topic (Röder, Both, and Hinneburg, 2015, p. 3).

C_V coherence is a composite measure that combines various statistical properties, including the indirect cosine measure with NPMI and a boolean sliding window. Although not elaborated upon in the referenced paper, it is found to perform well in various settings, indicating its adaptability

[3] `https://www.wikipedia.org`

and robustness (Röder, Both, and Hinneburg, 2015, p. 7).

2.6.4 Results of the First Level and Second Level Topic Allocation

The topic modeling process results in a set of distinct topics that encapsulate the underlying themes present in the dataset. These topics are manually entitled and briefly described to provide a clear understanding of the subject matter they represent. Each topic is characterized by a set of keywords generated through the LDA method, which identifies the words that are most indicative of the topic's content.

Furthermore, for each identified topic, the most representative paper is selected based on the highest probability score assigned by the LDA model. This score reflects the degree to which the paper aligns with the specific topic, providing a tangible example of the subject matter. The following sections present the entitled topics, their brief descriptions, and an overview of the most representative papers, offering a comprehensive insight into the thematic structure of the PdM domain.

Building on the aforementioned topic characterization, it is vital to evaluate the quality and coherence of the derived topics. The validation of the LDA model employed is visualized in Figure 5. Four prominent coherence metrics — C_V, UCI, UMASS, and NPMI — are used to gauge the semantic similarity between top words in each topic, thereby shedding light on topic interpretability (Röder, Both, and Hinneburg, 2015). An evident peak at $n = 6$ across the metrics accentuates the significance of this number of topics in the dataset. Specifically, the decline in C_V score post this mark hints at a possible oversaturation of topics, leading to diluted semantic value. Such observations drive the emphasis on judiciously choosing the number of topics, ensuring that the LDA model is both meaningful and discerning, especially in the sophisticated PdM domain (Blei, Ng, and Michael I. Jordan, 2003).

Traditional coherence metrics such as C_V, UCI, UMASS, and NPMI, while insightful, may present divergent evaluations as seen in this work's results (Doogan and Buntine, 2021; Harrando et al., 2021). To augment this, this research proposes a coherence measure grounded in the distribution of authors per paper. The underlying hypothesis suggests the existence of research clusters, indicated by authors predominantly contributing to a single topic (Xiong, M. Zhang, and Xing Wang, 2015; Contreras-Piña and Ríos, 2016). By analyzing the distribution of authors across topics, this measure aims to validate the number of topics generated by the LDA model. The rationale is that a well-defined topic should naturally attract a consistent group of authors,

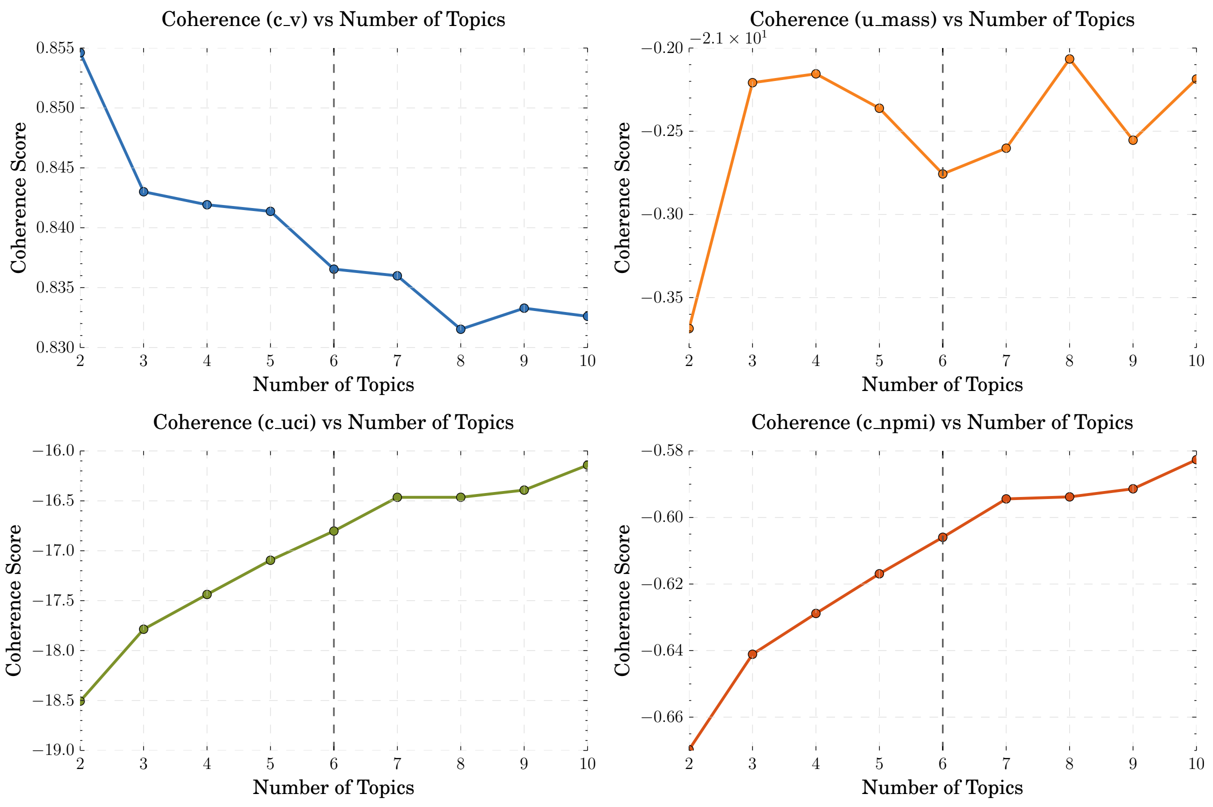


Figure 5: Comparison of different coherence scores vs number of topics: C_V in blue, UMASS in orange, UCI in green and NPMI in red (own figure).

thereby serving as an additional layer of validation for the coherence and relevance of the topics identified. This author-based coherence measure offers a complementary approach to existing metrics, providing a more nuanced understanding of topic quality and research clustering.

A polynomial function of degree 4 is fitted to this data, generating a curve that serves as a baseline for topic coherence. The choice of a fourth-degree polynomial is motivated by its ability to capture more complex, non-linear relationships in the data, which lower-degree polynomials may not adequately represent. The transformed data, obtained by subtracting the fitted values from the original average number of touched topics, serves as an additional layer of validation for the coherence and relevance of the topics identified. This approach aims to capture the clustering behavior of authors within topics, thereby providing a mathematical basis for evaluating topic quality.

Referencing Figure 6, a noticeable inflection at $n = 6$ in the original data (blue line) hints at its significance. The polynomial fit (in orange) effectively captures the data's nuances, and the pronounced deviations from the fit around $n = 6$ (grey bars) further emphasizes its importance.

Taking stock of these findings, the data suggests that the ideal number of topics is $n = 6$. Such a choice ensures semantically coherent topics that align with inherent research patterns in the

PdM domain.

In conclusion, the confluence of traditional coherence scores and the innovative author-based coherence metric solidifies the selection of $n = 6$ as the optimal number of topics for the PdM domain.

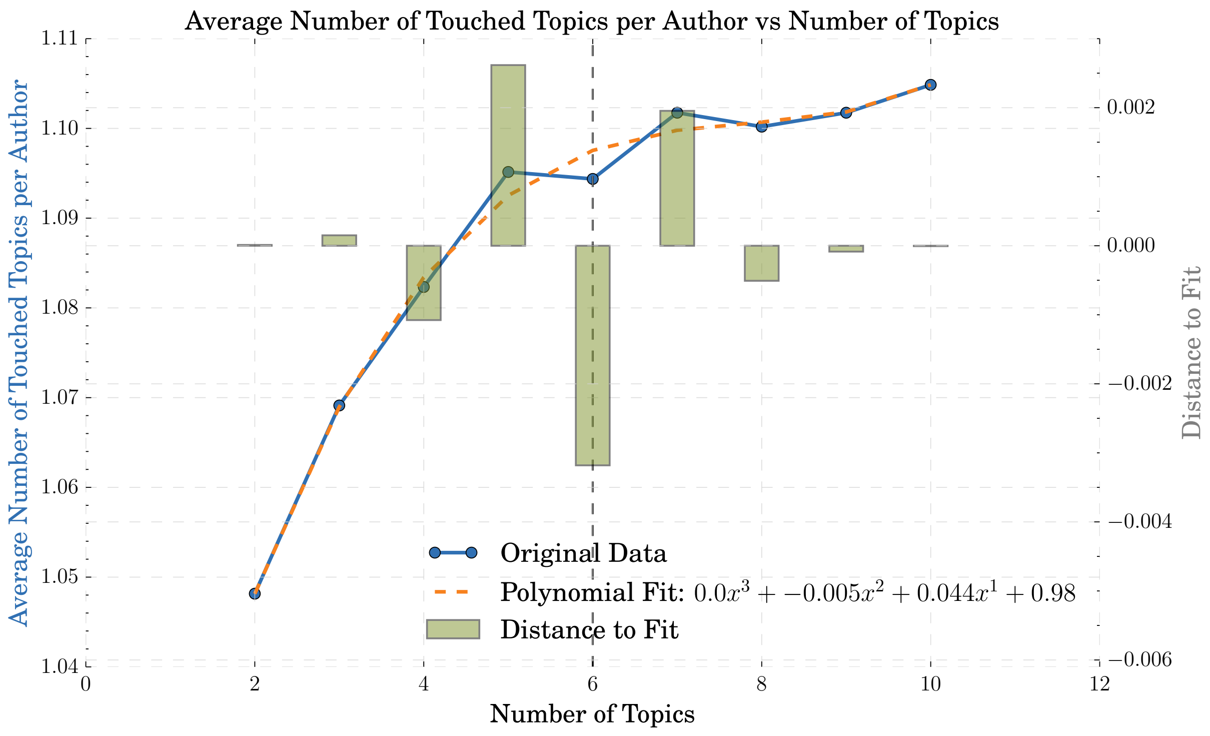


Figure 6: Average number of touched topics per author vs number of topics: Original data as blue line, polynomial fit as dotted orange line and distance to fit as green bars (own figure).

As depicted in Figure 7, the distribution of topics across the dataset appears to be well-balanced, lending further credence to the validity of the topic modeling approach (Maier et al., 2018). The relatively even distribution suggests that each topic captures a distinct facet of the PdM domain, without any single topic dominating the corpus (Gan and Qi, 2021). This observation substantiates the choice of $n = 6$ topics, reinforcing the notion that the model effectively partitions the data into meaningful clusters (B. Wang et al., 2014). Importantly, the distribution is not too equal, which could indicate random topic distributions, thereby further validating the model (Cao et al., 2009).

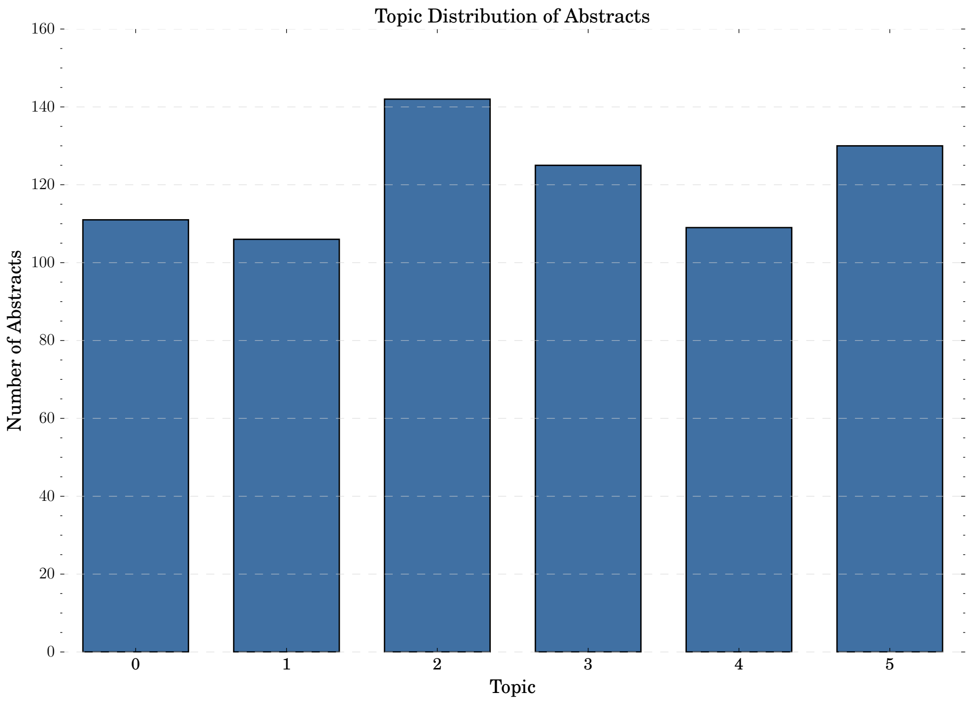


Figure 7: Number of abstracts that respectively included topic 1, topic 2, topic 3, topic 4, topic 5 and topic 6 (enumeration: 0 – 5) (own figure).

Analyzing the top keywords for each topic, as listed in Table 3, they are categorized into relevant domain-specific themes:

Topic	Top Keywords
Topic 1	[`fault`, `digital_twin`, `internet_of_things`, `robot`, `sensor`, `policy`, `network`]
Topic 2	[`model`, `failure`, `equipment`, `method`, `damage`, `artificial_neural_network`, `roughness`]
Topic 3	[`machine_learning`, `data`, `degradation`, `train`, `feature`, `gear_wear`, `cost`]
Topic 4	[`bearing`, `remaining_useful_life`, `prognostic`, `production`, `training`, `driven`, `portion`]
Topic 5	[`output`, `asset`, `manufacturing`, `portion`, `construction`, `review`, `risk`]
Topic 6	[`machine`, `battery`, `system`, `process`, `tool`, `scheduling`, `prognosis_health_management_system`]

Table 3: Topics 1 to 6 generated from topic modeling with their respective keywords.

- **Topic 1: IoT-Driven Robotic Systems**

This topic captures the integration of the Internet of Things (IoT) with digital twins and robotic systems. The emphasis on `fault`, `sensor`, `network`, and `policy` suggests a focus on fault detection and policy-driven decisions in IoT-enabled robotic systems. The convergence of these keywords reflects the growing importance of digitized and connected assets in modern industrial setups (H. Li et al., 2014). The work by H. Li et al. (2014) investigates the use of ML techniques and historical detector data to predict conditions leading to failure, thus avoiding service interruptions and increasing network velocity. H. Li et al. (2014) employ a range of analytical approaches, including ML techniques, to analyze historical detector data and various other types of data. Their models aim to predict conditions that could lead to future failures, thereby enhancing the reliability and efficiency of rail networks.

- **Topic 2: Equipment Damage Modeling**

Centered on `model`, `failure`, and `equipment`, this topic emphasizes the importance of understanding equipment damage and degradation. Keywords like `artificial_neural_network` and `roughness` hint at the use of advanced neural networks to model surface wear and tear, possibly in mechanical components (Cakir, Guvenc, and Mistikoglu, 2021). Cakir, Guvenc, and Mistikoglu (2021) discuss the advantages of IIoT systems that perform real-time monitoring, providing an early warning system for equipment failures. Cakir, Guvenc, and Mistikoglu (2021) developed an I4.0 compatible, IIoT-based Condition Monitoring System (CMS) that utilizes ML algorithms for real-time monitoring and failure prediction. Their system can send alerts to maintenance teams, thereby preventing severe equipment failures.

- **Topic 3: Machine Learning in Degradation Monitoring**

This topic predominantly revolves around ML techniques, with `machine_learning`, `data`, `degradation`, and `gear_wear` being the focal points. The inclusion of `cost` suggests the economic implications of machinery degradation and the benefits of early detection (Ewald et al., 2022). Ewald et al. (2022) propose a deep learning approach for structural health monitoring, emphasizing the need for *explainable AI* in this domain. They extend their previous work on deep learning for structural health monitoring by proposing a theoretical perspective inspired by neuroscience. They emphasize the need for *explainable AI* and discuss the challenges and opportunities of using deep learning for signal representation in structural health monitoring.

- **Topic 4: Bearing Life Prediction**

Emphasizing `bearing`, `remaining_useful_life`, and `prognostic`, this topic explores the estimation of a bearing's serviceable lifespan (Stollwitzer, Bettinelli, and J. Fink, 2023). Stollwitzer, Bettinelli, and J. Fink (2023) focus on the dynamic properties of the longitudinal track-structure interaction, highlighting the influence of various factors like vertical load and climatic conditions. Their findings suggest that traditional normative specifications may not adequately capture the complex behavior observed in their experiments.

- **Topic 5: Manufacturing Risk Management**

With `output`, `asset`, `manufacturing`, and `risk` as the primary keywords, this topic examines risk management in manufacturing processes (Garrido Martínez-Llop, Sanz Bobi, and Olmedo Ortega, 2023). Garrido Martínez-Llop, Sanz Bobi, and Olmedo Ortega (2023) discuss the use of ML to predict lateral car body accelerations, thereby enhancing passenger comfort and safety. They employ Long Short Term Memory Networks (LSTMs) to predict lateral car body accelerations, aiming to enhance passenger comfort and safety. Their model outperforms traditional ANNs and could be used for PdM, thereby reducing operational risks.

- **Topic 6: Machinery Health Prognosis Systems**

Highlighted by keywords such as `machine`, `battery`, `system`, and `prognosis_health_management_system`, this topic underscores the importance of machinery health management (Rokhforoz and O. Fink, 2021). Rokhforoz and O. Fink (2021) propose a hierarchical multi-agent framework for PdM scheduling in railway systems. Their approach uses dual decomposition and mechanism design to align the objectives of individual wagons with the central system, thereby optimizing maintenance scheduling and operational efficiency.

Following the initial LDA analysis, which segmented the literature into six broad topics, a second-level LDA analysis is thoroughly applied to each of these topics. This multi-level approach to topic modeling is innovative in its depth and specificity. While the first-layer LDA provides a macroscopic view of the thematic landscape, the second-layer LDA uncovers more nuanced subtopics within each primary category. Such an approach is particularly beneficial in complex fields like PdM, where a single layer of analysis may not suffice to capture the elaborate nuances and subthemes present in the literature. This layered analysis method, while not extensively documented, aligns with advanced topic modeling techniques discussed in recent studies (Blei, Ng, and Michael I. Jordan, 2003; Xiaolong Wang et al., 2011). By employing this multi-level LDA strategy, the research aims to construct a comprehensive and structured overview of the PdM field, aiding a deeper understanding of the literature and identifying potential areas for further investigation.

In addition to the detailed exploration of subtopics within each primary category, a significant aim of the second-level LDA analysis is to identify a unifying structure that interconnects the diverse main topics revealed in the first-layer analysis. This endeavor is driven by the principles of reusability central to CBSE. By uncovering commonalities and overarching themes that span across different main topics, the research seeks to establish a cohesive framework that embodies the CBSE ethos of modularity and reusability. Such a framework would not only enhance the understanding of the PdM domain but also advance the development of more efficient and adaptable PdM models. This approach aligns with the concept of reusability in software engineering, where identifying and leveraging common elements across different modules can significantly improve development efficiency and effectiveness (Krueger, 1992; Clements and

Northrop, 2002). This multi-level LDA analysis aims to reveal patterns and structures that can be applied universally across the PdM field, thereby embodying the reusability and modularity principles of CBSE.

Topic	Top Keywords
Topic 1: IoT-Driven Robotic Systems	
Subtopic 1.1:	[digital_twin, internet_of_things, network, sustainability, policy, independent_system_operator, lightning]
Subtopic 1.2:	[sensor, time_series, water, data, sector, parameter, caconvnet]
Subtopic 1.3:	[policy, network, dimension, training, running, underground, speed]
Subtopic 1.4:	[fault, industry_40, control, actuator, bearing, large, fuel]
Subtopic 1.5:	[stock, factor, reference_architecture, decision, joint, tso, twinbased]
Subtopic 1.6:	[robot, asset, varying, internet_of_things, input, corrosion, measure]
Topic 2: Equipment Damage Modeling	
Subtopic 2.1:	[failure, method, roughness, monitoring, aircraft, prediction, artificial_neural_network]
Subtopic 2.2:	[damage, aging, architecture, pipe, similaritybased_model, clinkering, learning]
Subtopic 2.3:	[model, equipment, artificial_neural_network, steel, company, operator, concept]
Topic 3: Machine Learning in Degradation Monitoring	
Subtopic 3.1:	[machine_learning, range, cost, capacitor, device, photovoltaic, fuzzy]
Subtopic 3.2:	[data, train, feature, gear_wear, opportunistic, autoencoder, artificial_neural_network]
Subtopic 3.3:	[degradation, algorithm, mechanical, group, failure, frequency, artificial_intelligence_machine-_learning]
Topic 4: Bearing Life Prediction	
Subtopic 4.1:	[leased, layer, technology, data, result, led, vibration]
Subtopic 4.2:	[bearing, training, singlelayer_reticulated_shells, module, damping, ballasted, inflection]
Subtopic 4.3:	[prognostic, alert, layer, induction_motor, line, zno_arrester, tool]
Subtopic 4.4:	[production, service, vibration, endofline_testing, underground_infrastructure, industrial, digital]
Subtopic 4.5:	[remaining_useful_life, portion, driven, rolling_bearing, indicator, system, lifetime]
Topic 5: Manufacturing Risk Management	
Subtopic 5.1:	[review, mechanical, tunnelling, smart_factory, technique, permutation_entropy, mp]
Subtopic 5.2:	[accident, construction, strategy, transport, prepipe, connection, sensor]
Subtopic 5.3:	[portion, risk, failure, displacement, result, lithiumion_battery, electrical]
Subtopic 5.4:	[output, planning, barrier, spm_practice, task, nbr, seal]
Subtopic 5.5:	[manufacturing, construction, gearbox, air, tunnelling, control, building_information_modeling]
Subtopic 5.6:	[asset, condition, pattern, aim, aircraft, h_division, test]
Topic 6: Machinery Health Prognosis Systems	
Subtopic 6.1:	[tool, planning, scheduling, energy, break, prognosis_health_management_system, machine_learning]
Subtopic 6.2:	[process, plant, breakdown, business, line, thermal_image, power]
Subtopic 6.3:	[system, battery, cross, condition, charge, residential, approach]
Subtopic 6.4:	[machine, generator, oil, methodology, industry_40, conditionbased_maintenance, dock]

Table 4: Topics 1 to 6 generated from topic modeling and their respective subtopics with keywords.

The outcomes of the second-level LDA analysis are concisely presented in Table 4, which enumerates the subtopics identified within each of the six main topics derived from the initial LDA analysis. In determining the number of subtopics for each main topic, the methodology mirrored that of the initial analysis. This approach ensured consistency and methodological rigidity across both levels of analysis. The number of subtopics per main topic varied, ranging from three to six. This variation reflects the inherent complexity and depth of each main topic, with some topics exhibiting a broader range of subthemes than others. The decision on the number of subtopic is guided by the goal of achieving a comprehensive yet coherent categorization, balancing the need to capture the diversity within each main topic against the risk of over-fragmentation. This methodical approach to topic selection is rooted in established practices in topic modeling, ensuring that the identified subtopics are both representative and meaningful within the context of the broader PdM research landscape (Blei, Ng, and Michael I. Jordan, 2003; Griffiths and Steyvers, 2004).

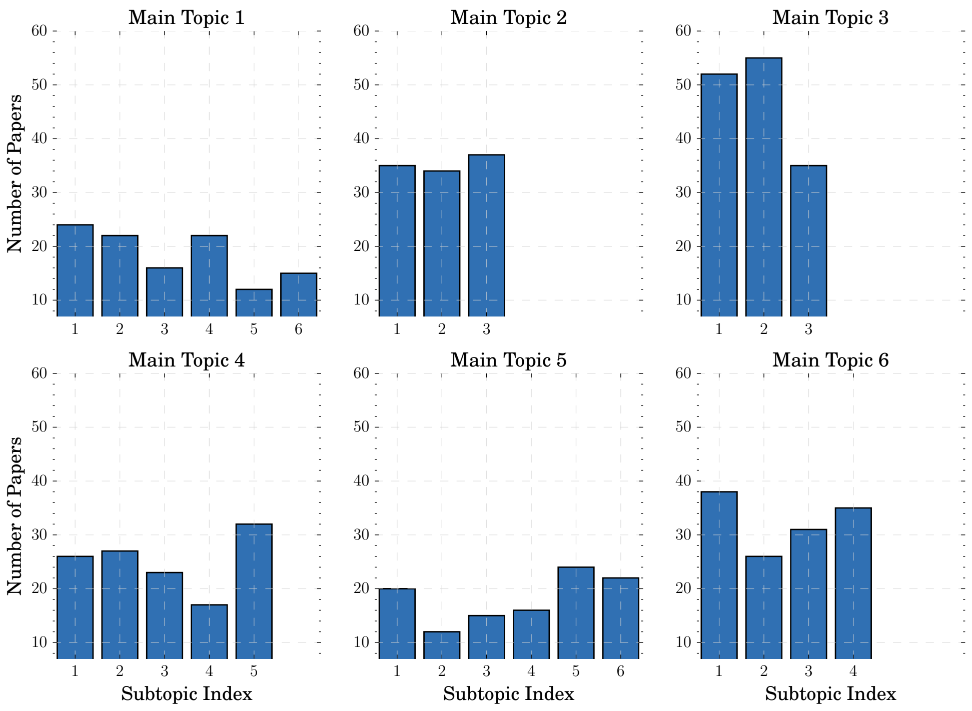


Figure 8: Number of papers per main topic and respective subtopics (own figure).

The distribution of scientific papers across the subtopics, as depicted in Figure 8, provides valuable insights into the thematic spread within the PdM literature. The figure illustrates a relatively balanced but not uniform distribution of papers among the subtopics. This pattern of distribution is indicative of a well-calibrated LDA model, as it suggests a comprehensive coverage of the thematic landscape without any single subtopic dominating the corpus. The absence of a skewed distribution towards any particular subtopic implies that the LDA model successfully avoids overfitting, a common concern in topic modeling where the model might overly specialize in certain topics at the expense of others. This balanced representation ensures that the model captures a wide array of themes and perspectives within the PdM field, thereby providing a holistic view of the research landscape. Such a distribution is vital for ensuring that the subsequent analysis and interpretations are grounded in a diverse and representative sample of the literature, thereby enhancing the reliability and validity of the findings.

The comprehensive table of subtopics 4 reveals a structured pattern across the major topics in the PdM domain. Each topic appears to have recurring elements that can be categorized into four main aspects: data source, feature engineering, modeling technique, and modeling purpose. For instance, Topic 1, *IoT-Driven Robotic Systems*, includes subtopics that touch upon data sources like `digital_twin` and `internet_of_things`, as well as modeling techniques such as `network` and `caconvnet`[4] (K. Lee et al., 2017). In Topic 2, *Equipment Damage Modeling*, `equipment` and `pipe` serve as data sources, and `failure` and `prediction` align with the modeling purpose. This pattern is consistent across other topics, reinforcing the idea that a structured approach to PdM is beneficial for both research and practical applications (March and Scudder, 2019; Cheng et al., 2020). The recurring elements in the subtopics validate the need for a multi-disciplinary approach in PdM, encompassing diverse aspects from data collection to Model Deployment (MD) (Achouch et al., 2022; Chuang et al., 2019).

Category	Keywords
Data Source	
	[sensor, monitoring, device, equipment, pipe, photovoltaic, train, aircraft, lithiumion_battery, data, underground_infrastructure, generator, internet_of_things, vibration, building_information_modeling, rolling_bearing, mechanical, digital_twin, damping, bearing, battery]
Feature Engineering	
	[control, condition, algorithm, corrosion, indicator, tool, displacement, method, frequency, dimension, measure, degradation, result, feature, input, time_series, output, factor, pattern, thermal_image, gearbox]
Modeling Technique	
	[artificial_intelligence_machine_learning, fuzzy, learning, machine_learning, artificial_neural_network, autoencoder, permutation_entropy, model, layer, network, training, technique, similaritybased_model, caconvnet]
Modeling Purpose	
	[remaining_useful_life, aging, accident, sustainability, lifetime, induction_motor, damage, prognostic, failure, prediction, decision, risk, prognosis_health_management_system, breakdown, aim, fault, break, range, gear_wear]
Unclassified	
	[smart_factory, policy, ballasted, underground, operator, group, test, clinkering, oil, independent_system_operator, manufacturing, strategy, transport, prepipe, zno_arrester, barrier, spm_practice, led, business, machine, approach, planning, line, fuel, inflection, company, portion, cross, varying, architecture, capacitor, joint, system, asset, technology, task, singlelayer_reticulated_shells, production, residential, opportunistic, endofline_testing, energy, running, h_division, industrial, stock, cost, process, power, seal, dock, roughness, sector, reference_architecture, scheduling, air, service, charge, concept, nbr, parameter, large, industry_40, review, mp, connection, alert, module, tso, tunnelling, robot, plant, leased, construction, electrical, water, actuator, digital, driven, speed, steel, lightning, conditionbased_maintenance, twinbased, methodology]

Table 5: Subtopic keywords classified into the categories *Data Source*, *Feature Engineering*, *Modeling Technique*, *Modeling Purpose*, and *Unclassified.*

In an effort to further refine the understanding of the subtopics identified through the second-

[4]Causal Augmented Convolution Network

level LDA analysis, each keyword extracted from these subtopics is thoroughly classified and is presented in Table 5. The classification scheme employed categorizes keywords into the four distinct phases of PdM model development: *Data Source*, *Feature Engineering*, *Modeling Technique* and *Modeling Purpose.* This categorization is instrumental in illustrating the specific aspects and stages of PdM model development that each keyword relates to, thereby providing a clearer picture of the thematic focus within each subtopic. Additionally, certain keywords that are either ambiguous in nature or not directly relevant to the specific phases of PdM model development are labeled as *Unclassified.* This distinction is key for maintaining the clarity and relevance of the categorization, ensuring that only keywords with a direct and clear association to the phases of PdM model development are included in the respective categories. This manual classification process not only aids in the thematic organization of the keywords but also serves as a foundational step in identifying potential areas for the application of CBSE principles within the PdM model development process.

The classification of keywords derived from the subtopics into distinct categories, as depicted in Table 5, offers insightful perspectives into the various facets of PdM model development. In the *Data Source* category, keywords such as `sensor`, `vibration`, and `digital_twin` emphasize the diverse range of equipment and technologies involved in data collection for PdM. For instance, `sensor` and `vibration` highlight the importance of monitoring physical parameters, while `digitaltwin` represents the advanced digital modeling of physical systems. The *Feature Engineering* category, with keywords like `algorithm`, `time_series`, and `thermal_image`, underscores the methods and tools used to process and analyze data, where `algorithm` suggests computational techniques, and `thermal_image` points to specific data types used for analysis. In the *Modelling Technique* category, terms such as `artificial_neural_network` and `machine_learning` reflect the advanced computational methods employed in PdM, indicating a strong focus on AI and ML techniques. The *Modelling Purpose* category includes keywords like `remaining_useful_life` and `prognostic`, which are directly related to the objectives of PdM models, such as predicting equipment lifespan and anticipating failures. Lastly, the *Unclassified* category contains a wide array of keywords like `smart_factory` and `industry_40`, which, although relevant to the broader context of PdM, do not directly align with the specific phases of PdM model development. This categorization not only aids in structuring the vast array of keywords but also provides a clear thematic mapping of the PdM literature, reflecting the multifaceted nature of this field.

Figure 9 presents a compelling visualization of the distribution of keywords across different phases

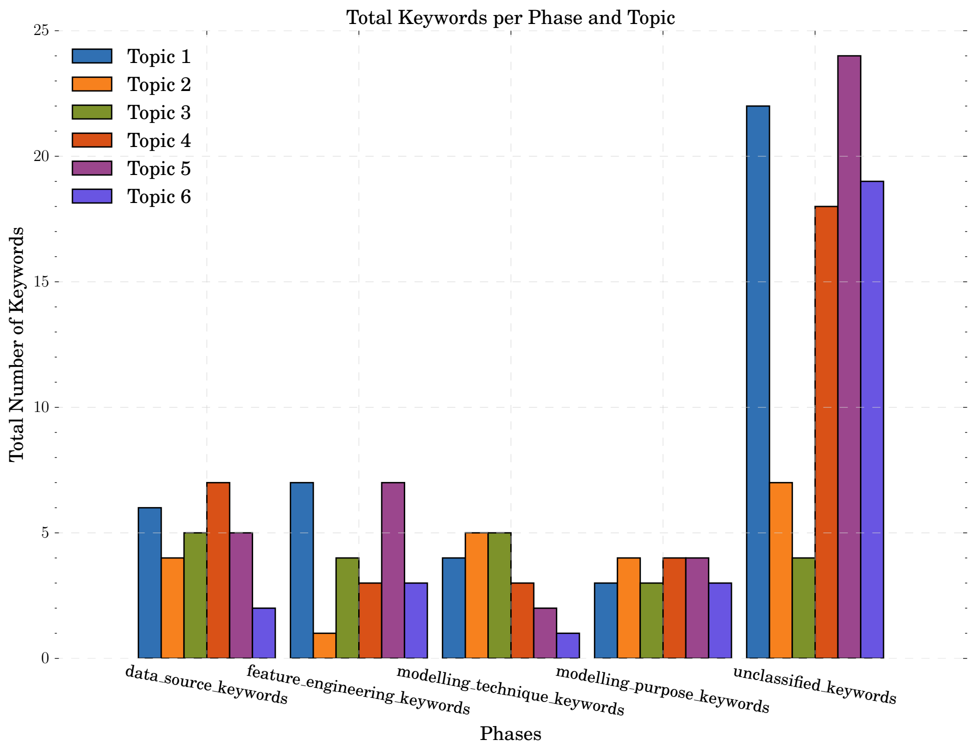


Figure 9: Number of keywords per topic 1 to 6, divided into the categories *Data Source*, *Feature Engineering*, *Modeling Technique*, *Modeling Purpose*, *Unclassified* (own figure).

of PdM for each subtopic identified in the literature. Notably, every phase is represented in each subtopic, at least to some extent. This uniformity in representation across phases underscores the comprehensive nature of the PdM field, as it suggests that each aspect of PdM is integral to the understanding of every subtopic. The similarity in distribution patterns across different subtopics further reinforces the hypothesis that there exists a fundamental structure underlying all PdM topics. This structure is characterized by a consistent emphasis on all phases of PdM, from data sourcing to modeling techniques and purposes. Such a finding is significant as it indicates that, despite the diversity of themes and focus areas within PdM literature, there are core elements that are universally pertinent. This revelation not only validates the methodological approach employed in this research but also provides a foundational understanding that can be instrumental in guiding future explorations and developments in the field of PdM.

The detailed distribution of keyword appearances across individual subtopics, as illustrated in Figure 10, offers insightful observations into the thematic structure of PdM. While there is some overlap in keyword appearances, particularly noticeable in subtopics like Topic 5, a discernible separation between phases such as *Modeling Technique* and *Feature Engineering* is evident. This separation is indicative of the distinct nature of these phases within the PdM process. In Topic 1, for instance, the keyword concentrations are distributed across all phases, demonstrating a

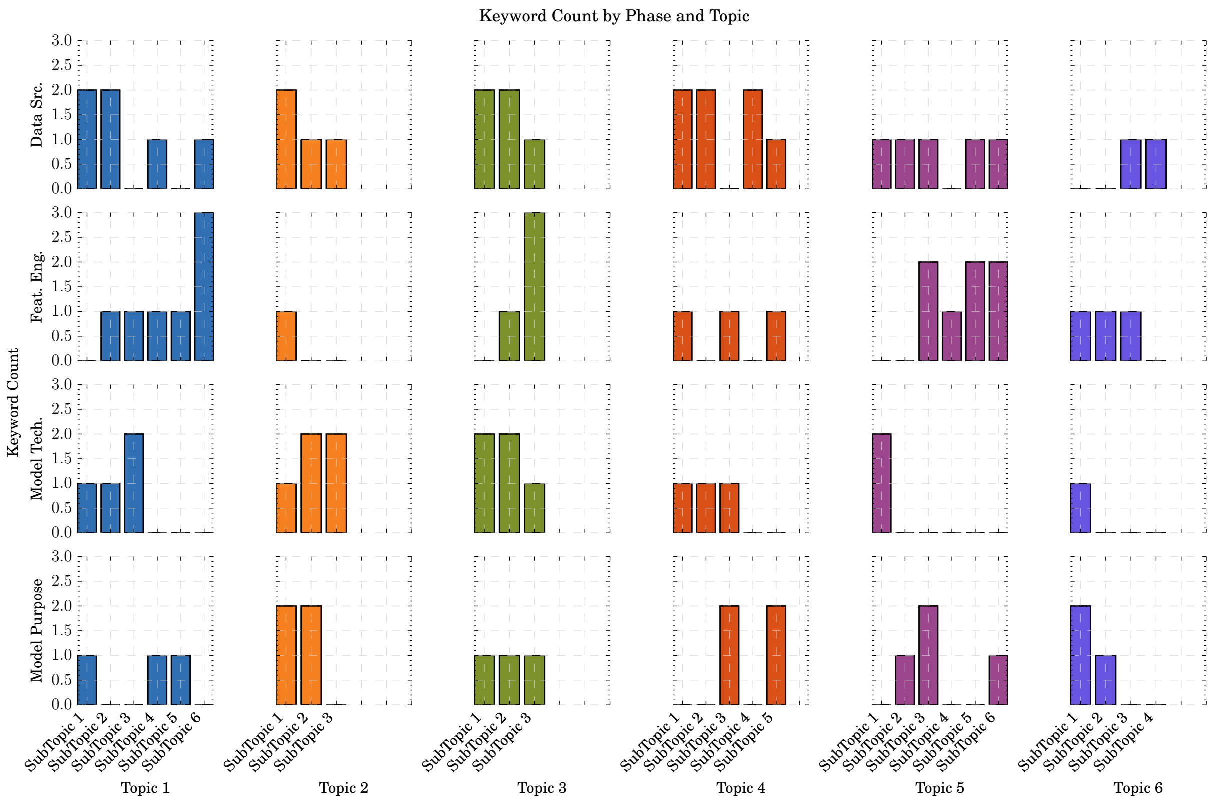


Figure 10: Keyword count divided into keyword categories, per topic and respective subtopics (own figure).

comprehensive coverage of the PdM spectrum. Overall, the distribution patterns observed in the subtopics, as derived from the LDA analysis, reflect the anticipated phases of PdM. This alignment between the LDA-derived subtopics and the hypothesized phases of PdM suggests that the LDA model effectively captures the inherent structure of the PdM field. The ability of the LDA model to show these phases within the subtopics not only validates the analytical approach but also highlights the nuanced complexity of PdM, where different aspects intertwine yet maintain distinct identities. Such findings are instrumental in understanding the multifaceted nature of PdM and in guiding future research to explore these phases in more depth, thereby contributing to the advancement of the field.

In the context of PdM, the classification of keywords into distinct phases is fundamental for a structured approach to model development and implementation. Table 6 encapsulates this classification effectively.

Data Source: This category includes keywords such as `sensor`, `vibration`, and `bearing`, highlighting the importance of sensors in collecting real-time data, a fundamental aspect of PdM (Randall, 2011). Additionally, `device`, `aircraft`, and `internet_of_things` reflect the diverse sources of log data, essential for predictive analytics in various industrial applications

Category	Keywords
Data Source	
Sensor	[sensor, monitoring, vibration, bearing, battery, digital_twin, mechanical, rolling_bearing, damping]
Log (Events)	[device, equipment, pipe, photovoltaic, train, aircraft, lithiumion_battery, data, underground_infrastructure, generator, internet_of_things, building_information_modeling]
Feature Engineering	
Statistical	[control, condition, corrosion, indicator, displacement, frequency, dimension, measure, degradation, result, feature, input, output, factor]
Pattern / Time / Frequency-based	[time_series, thermal_image, pattern]
Static	[tool, method, gearbox]
Modeling Technique	
Model-based	[model, technique, similaritybased_model, caconvnet, permutation_entropy]
Data-based	[artificial_intelligence_machine_learning, layer, network, training, fuzzy, learning, machine_learning, artificial_neural_network, autoencoder]
Modeling Purpose	
RUL Prediction	[remaining_useful_life, lifetime, prognostic, prediction, prognosis_health_management_system, range, gear_wear, induction_motor]
Deviation Detection	[aging, accident, damage, failure, fault, break]
Maintenance Need Classifier	[sustainability, decision, risk, breakdown, aim]

Table 6: Classification per keywords per category into PdM phases (own figure).

(García Márquez et al., 2012).

Feature Engineering: Keywords like `corrosion` and `degradation` under the statistical subcategory indicate the use of statistical methods to identify significant features for equipment health assessment (Jardine, Lin, and Banjevic, 2006). The inclusion of `time_series` and `pattern` under pattern/time/frequency-based feature engineering underscores the importance of analyzing temporal patterns in PdM (Sikorska, Hodkiewicz, and Ma, 2011).

Modeling Technique: The model-based technique, with keywords such as `model` and `similaritybased_model`, suggests the use of physical or mathematical models in PdM (Si et al., 2011). In contrast, the data-based technique, indicated by `machine_learning` and `artificial_neural_network`, points towards the growing role of AI and ML in PdM (R. Zhao et al., 2016).

Modeling Purpose: The purpose of modeling in PdM is well-represented with keywords like `remaining_useful_life` and `prognostic` for Remaining Useful Life (RUL) prediction, emphasizing the goal of predicting machinery's RUL (Si et al., 2011). Keywords such as `failure` and `fault` in deviation detection highlight the need for early fault detection, a key component in PdM (Jardine, Lin, and Banjevic, 2006). Lastly, maintenance need classifier keywords like `risk` and `decision` reflect the decision-making processes in maintenance based on risk assessment

(W. Wang, 2012).

This classification not only aids in understanding the various facets of PdM but also serves as a guide for selecting appropriate methodologies and techniques in the development of PdM models.

2.6.5 Validation of the Topic Results

Figure 11 presents a comprehensive framework for PdM, as a culmination of the research efforts detailed in this work. This framework, initially introduced by Fromm (2020), is quantitatively validated and further refined through an extensive two-layer LDA analysis conducted as part of this study. The framework is segmented into distinct phases, each representing a critical component in the PdM process.

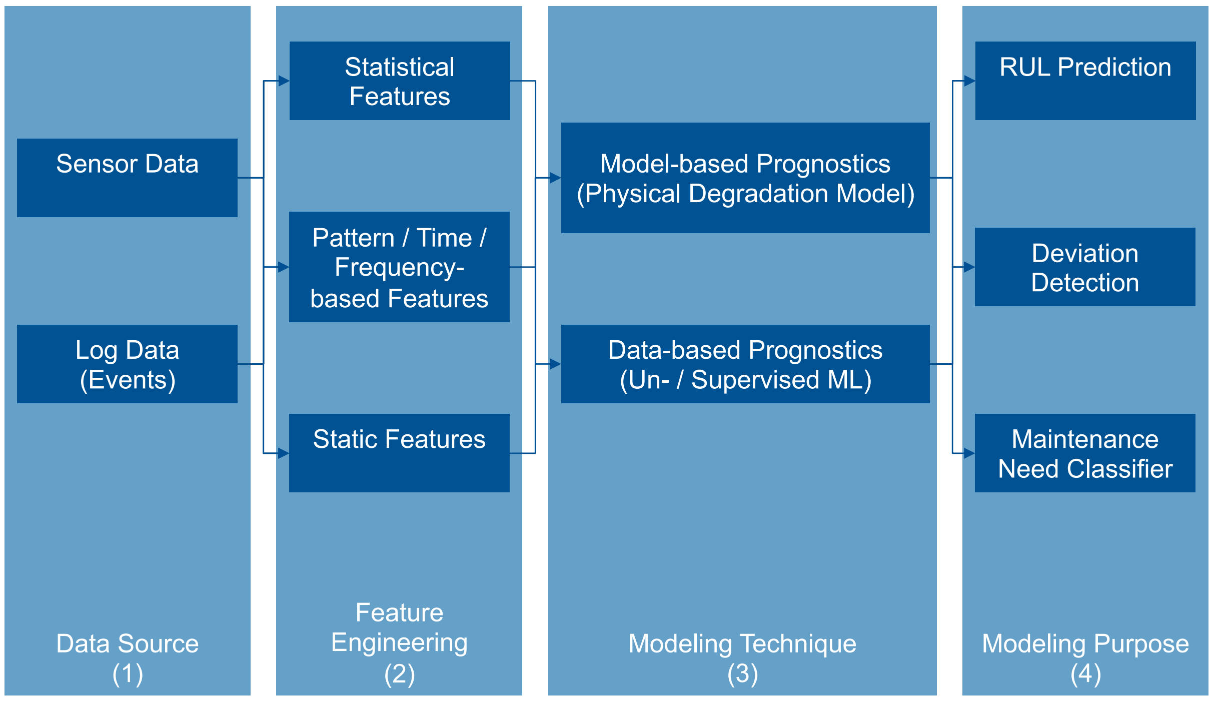


Figure 11: *Predictive Maintenance Framework* Framework divided into four PdM phases as a result of two-layer dirichlet Allocation, validated by Fromm (2020).

The validation of the PdM framework developed in this study is underpinned by a thorough analysis conducted by the author in an unpublished thesis.

This analysis involved a manual examination of a representative set of publications, distinct from the initial dataset, to identify the presence and relevance of the framework's phases and classifications within these works. The findings of this analysis are systematically presented in Table 7, as included in the supplementary material. The table illustrates how the various elements of the PdM framework are consistently mirrored across a broad spectrum of publications

Phase	1		2			3		4		
Article	**Sensor**	**Log**	Statistical	Patterns	Static	**Model-driven**	**Data-driven**	RUL	Deviation	Maintenance Need
N. Gebraeel, Elwany, and Jing Pan (2009)	•	•	•				•	•		
Balali, Seifoddini, and Nasiri (2020)	•	•		•			•	•		
Herpel et al. (2009)	•		•	•		•			•	
Eke et al. (2017)	•		•	•		•			•	
Staszewski, K. Worden, and Tomlinson (1997)	•		•	•			•		•	
Farrar and Keith Worden (2012)	•		•	•			•		•	
Bunks, Mccarthy, and Al-Ani (2000)	•		•		•	•		•		
Mann, Saxena, and Knapp (1995)	•		•			•	•		•	
Gašperin et al. (2011)	•		•			•		•		
Fugate, Sohn, and Farrar (2001)	•		•				•		•	
Pan, Sas, and Van Brussel (2003)	•		•				•			•
N. Gebraeel (2006)	•			•	•		•	•		
Goode, Moore, and Roylance (2000)	•			•		•		•		
Sipos et al. (2014)		•	•	•	•		•		•	•
Bin Hu et al. (2012)		•	•	•			•	•		•
J. Wang et al. (2017)		•	•	•			•		•	•
S. He et al. (2016)		•	•	•			•		•	
Patil et al. (2017)		•	•	•			•			•
M.-L. T. Lee and Whitmore (2006)		•	•		•		•	•		
Gutschi et al. (2019)		•	•		•		•		•	
Shimada and Sakajo (2016)		•	•				•			•
Decker et al. (2020)		•		•	•	•			•	•
Serna et al. (2011)		•		•	•	•				•
Rögnvaldsson et al. (2018)		•		•	•		•		•	

Table 7: Overview of the Aspects of each PdM Phase that are covered in each publication of a representative set of PdM literature.

within the PdM domain.

In Table 7, it becomes apparent that the framework's components are not only applicable but also prevalent in diverse industrial research contexts. While the selection of publications for closer examination in this review is not exhaustive, it is intentionally diverse, offering a comprehensive overview of the industrial facets pertinent to PdM research.

Further deepening the analysis, Table 8 presents a quantitative assessment of the interrelationships between different aspects of PdM. This table showcases a matrix where the frequencies of combinations between two aspects $a \in A$ of PdM are calculated. Normalized against the set of reviewed publications $p \in P$, the matrix reveals the observation frequencies $x_{m,n}$, mathematically rounded to one decimal place, for each aspect tuple defined by the row and column indices m and n. Notably, the main diagonal of the matrix indicates the proportional observation frequency x_a of each individual aspect. This quantitative approach provides a nuanced understanding of how various elements of PdM interconnect and co-occur within the research landscape, offering valuable insights into the multifaceted nature of PdM in industrial applications.

Phase	1		2			3		4			
Aspects	**Sensor**	**Log**	Statistical	Patterns	Static	**Model-driven**	**Data-driven**	RUL	Deviation	Maintenance Need	**TOTAL**
Sensor	.5	.1	.4	.3	.1	.3	.3	.3	.3	.0	
Log		.5	.4	.4	.3	.1	.5	.2	.3	.3	
Statistical			.8	.4	.2	.2	.6	.2	.4	.3	
Patterns				.6	.2	.2	.4	.2	.4	.3	
Static					.3	.1	.2	.1	.2	.1	
Model-driven						.3	.0	.1	.2	.1	
Data-driven							.7	.2	.4	.3	
RUL								.3	.0	.0	
Deviation									.5	.1	
Maintenance Need										.3	
TOTAL	**13**	**13**	**18**	**15**	**8**	**8**	**17**	**8**	**12**	**8**	**24**

Table 8: Result matrix of the frequency analysis of the aspects of each PdM phase in a representative set of PdM literature, showing the observation frequencies rounded to one decimal place.

$$x_{m,n} = \frac{|\{p \in P : a_m, a_n \in p\}|}{|P|}, \ \forall m, n \in A \tag{2.6.7}$$

The validation of the PdM framework, as depicted in Figure 11, is substantiated through a frequency analysis of various aspects within the PdM domain. This analysis, detailed in Table 8, demonstrates a comprehensive coverage of the considered articles across the defined data types, with a balanced distribution between sensor and log data. This balance justifies the bifurcation into these two data types within the scope of this research. *Sensor data*, encompassing real-time system monitoring data streams, emerge as the primary data source in numerous technical systems, influencing the choice of feature engineering methods. *Log data*, on the other hand, are event-driven and often preferred due to their digital format and dependency on primary sensor data. This phase is essential as approximately 80% of the development effort in PdM is invested in exploratory analysis and data cleansing (Dasu and Johnson, 2003).

The second phase, *Feature Engineering*, shows a complete representation of the aspects *Statistical*, *Patterns*, and *Static* in the reviewed literature. However, static feature engineering methods are mentioned in only about 30% of the publications, suggesting their often implicit role in the research process. The high overlap of these aspects indicates a potential for parallelization in feature engineering methods, which is vital for the modular design introduced in Section 4. The combination of feature engineering aspects with data sources reveals a tendency to discuss static methods predominantly when log data are used, highlighting the complexity of preprocessing and transforming event-triggered features.

In the third phase, *Modeling Technique*, the literature commonly differentiates between data-based and model-based (or physics-driven) approaches. The analysis reveals a dominance of data-driven models, with about 70% of the publications discussing these approaches. However, this does not diminish the importance of model-based approaches, which often implicitly influence data-driven models, especially in feature selection based on expert recommendations or physical correlations. The frequent use of statistical feature engineering methods in data-driven predictions underscores the importance of data quality and volume in enhancing model performance.

Finally, the *Modeling Purpose* phase is categorized into *RUL Prediction*, *Deviation Detection*, and *Maintenance Need Classifier*. The RUL prediction, a key factor in Condition Monitoring (CM), is primarily achieved through supervised ML regression methods, focusing on predicting the RUL based on usage and condition monitoring (Si et al., 2011). Deviation detection, often employing unsupervised ML methods, monitors outliers and irregularities, suitable for systems with indefinite lifespans or minimal failure data. The Maintenance Need Classifier, typically using supervised ML classification algorithms, assesses the current or future state of a system for maintenance requirements. The literature review indicates a slight preference for Deviation Detection models, with a notable observation that sensor data are rarely combined with Maintenance Need Classification, possibly due to the categorical nature of event-based log data being more conducive to classification approaches than to regression models.

2.7 Conclusion of the Literature Findings

The topic modeling process reveals a rich and multifaceted landscape in the domain of PdM. By identifying eight distinct topics, each characterized by a set of keywords and represented by a paper with the highest probability score, the complexity and breadth of the field are clearly demonstrated.

This section synthesizes the findings from the literature review, addressing the research questions and critically analyzing the strengths and weaknesses of the methodologies employed. The systematic exploration conducted herein validates the hypothesis that PdM fits the requirements of CBSE, indicating that modularization and reusability bring significant benefits to the development of PdM models. However, it also highlights a research gap in the thorough application of CBSE principles to PdM, which this work aims to address.

The findings from various subsections of this literature review demonstrate that while PdM models benefit from CBSE approaches, there is a lack of comprehensive methodologies that fully exploit the potential of these principles. The empirical studies reviewed suggest that while components of CBSE are often applied to specific facets of PdM, a holistic and integrated approach is still missing. This gap presents a significant opportunity for future research to develop a more cohesive and standardized framework that can pave the way for the efficient and effective implementation of PdM systems in industrial applications.

In conclusion, this literature review lays a solid foundation for the proposed component repository, highlighting both the necessity and potential for leveraging CBSE in PdM. The subsequent phases of this thesis will build upon these findings, aiming to design and validate a novel framework that not only addresses the identified gaps but also enhances the scalability, reusability, and efficiency of PdM models.

Chapter 3

Methods

This chapter articulates the comprehensive methodological framework employed in this dissertation, which underpins the systematic development and evaluation of artifacts designed to advance the field of PdM. Central to this work is the adoption of DSR, recognized for its robustness in iteratively creating and refining artifacts that address complex real-world problems. Moreover, this chapter introduces the specific research environment which plays a critical role in shaping the research outcomes. The automotive industry, characterized by rapid technological innovations and high data availability, provides a fertile ground for deploying and testing the developed PdM models, thus offering practical insights into their scalability and effectiveness. Additionally, this chapter examines the ethical considerations inherent in this research, ensuring that the methodologies and artifacts not only adhere to technical standards but also align with broader ethical norms and values. These aspects are crucial for ensuring that the solutions developed are not only technically sound but also socially responsible and acceptable.

3.1 Design Science Research as Research Design

In this thesis, DSR is employed as the primary research design to address the stated research challenges. DSR, a methodology renowned for its iterative process and artifact creation, is a fitting approach given the exploratory and innovative nature of the research goals. The foundation of DSR rests on creating and evaluating artifacts, with the objective of solving real-world problems and adding to theoretical knowledge. Throughout the study, various artifacts, including models, methods, and prototypes, are developed and refined in multiple iterations. These iterations allow for continuous feedback and improvement, ensuring the relevance and applicability of the solutions presented. The use of DSR not only provides a structured and stringent methodological framework but also allows the bridging of the gap between theoretical research and practical application.

DSR is a methodology used to create and evaluate IT artifacts intended to solve identified organizational problems. DSR is particularly relevant in the field of PdM, where innovative solutions can significantly enhance operational efficiency and reliability. This research adopts DSR due to its potential for practical contribution and its alignment with the objectives of developing a PdM component repository (Hevner et al., 2004).

The principles of DSR are vital to the development of artifacts that are innovative, relevant, thoroughly evaluated, and contribute to knowledge. These principles underpin the research process, ensuring that the artifacts produced are technologically robust and add meaningful insights to the existing body of knowledge.

Innovative artifact creation is the essence of DSR, focusing on the development of new models, constructs, methods, or instantiations that effectively address identified problems (Gregor and A. R. Hevner, 2013). The relevance to real-world problems is highly important, as DSR seeks to solve practical issues, ensuring the research's impact and applicability (Venable, Pries-Heje, and Baskerville, 2012). Stringent evaluation is integral, requiring systematic and empirical assessment of the artifact's utility, quality, and efficacy within its domain (Peffers et al., 2007). Lastly, the contribution to knowledge is a fundamental outcome of DSR, where the research should extend the theoretical foundations and offer guidance for future research and practice (Hevner et al., 2004).

Empirical research methodologies and DSR differ fundamentally in their approach, objectives, and outcomes. Empirical research is primarily descriptive and explanatory, aiming to understand and interpret phenomena through observation and analysis. It seeks to establish patterns, relationships, or theories based on empirical evidence, often employing statistical methods to analyze data (Yin, 2018).

In contrast, DSR is inherently prescriptive and constructive. It is concerned with creating and evaluating artifacts designed to address specific problems. DSR does not just seek to understand the world but to actively change it by introducing new and innovative solutions. The outcomes of DSR are tangible artifacts—models, methods, constructs, or instantiations—whereas empirical research contributes through the validation of hypotheses and enhancement of theoretical frameworks (Hevner et al., 2004; March and Smith, 1995).

The objectives of empirical research are often to test theory or to extend understanding of a given phenomenon, which can lead to generalizable knowledge. DSR, on the other hand, is goal-oriented towards the creation of effective solutions, with a stronger emphasis on practical relevance and applicability. The evaluation in DSR is centered on the artifact's performance in the real world, while empirical research focuses on the robustness of the theoretical insights derived from data (Simon, 2008; Sein et al., 2011).

In summary, while empirical research contributes to the understanding of 'what is', DSR is directed towards 'what could be', providing a proactive pathway to innovation and improvement in the field of Information Systems (IS) and beyond.

Action research and DSR share a common iterative approach but diverge significantly in their focus and execution. Action research is a participatory process that involves stakeholders at every stage, emphasizing collaborative problem-solving, planning, action, and reflection within a cycle of continuous improvement (Lewin, 1946). It is characterized by its dual commitment to bringing about change in the real world and to generating new knowledge, with the researcher often acting as a facilitator within the organizational context (Reason and Bradbury, 2001).

DSR, while it may engage stakeholders, does not inherently require their participation in the same manner. The primary goal of DSR is the creation of innovative artifacts, and while stakeholder input may inform the design process, it is the artifact itself that is central to the methodology. The iterative cycles in DSR focus on the refinement and evaluation of the artifact, rather than on the participatory action and reflection cycles of action research (Hevner et al., 2004; Peffers et al., 2007).

Furthermore, action research is often context-bound, with findings and solutions tailored to specific organizational settings. In contrast, DSR aims to produce artifacts that have broader applicability and can be generalized beyond the initial context in which they are created. The participatory nature of action research can lead to rich, contextually grounded knowledge, but it may limit the generalizability of the results. DSR, on the other hand, seeks to contribute to a body of knowledge that can inform the design and application of artifacts across various contexts (Simon, 2008; Sein et al., 2011).

In essence, while action research is a method to enact change collaboratively and learn from the process within a specific context, DSR is a method to create and evaluate generalizable solutions that can be applied across multiple contexts, with less emphasis on the participatory aspect of problem-solving.

Case study research stands in contrast to DSR in its methodological approach and objectives. It is an in-depth investigation of a particular individual, group, or event, aimed at understanding the dynamics present within single settings (Yin, 2018). Case studies are often used in exploratory research where the boundaries between the phenomenon and context are not clearly evident, and they allow researchers to retain the holistic and meaningful characteristics of real-life events.

DSR, in contrast, is not confined to the detailed examination of a specific context but is oriented towards the creation of artifacts that can be applied in a variety of contexts. While case study research provides a depth of understanding and richness of detail, it often lacks the generalizability that DSR seeks to achieve through the development and evaluation of prototypes or models (Hevner et al., 2004; Venable, Pries-Heje, and Baskerville, 2012).

The goal of case study research is to arrive at a comprehensive understanding of the event or entity being studied, which may include generating theories from the data. DSR, however, is inherently constructive, aiming to produce new, innovative solutions to problems. The artifacts developed in DSR are intended to be generalizable and transferable, offering solutions that extend beyond the specific case from which they arose (Gregor and A. R. Hevner, 2013; Vaishnavi, Vaishnavi, and Kuechler, 2015).

To sum up, while case study research focuses on understanding phenomena in their real-life context with an emphasis on depth over breadth, DSR is concerned with the creation of artifacts that can be generalized and applied across various contexts, prioritizing breadth of application and the potential for innovation.

The methodology of DSR employed in this thesis is characterized by a series of distinct yet interrelated stages that collectively contribute to the development of effective, innovative solutions in the realm of PdM. Each stage is described in detail below:

Problem Identification and Motivation: The DSR journey begins with a thorough understanding of the problem at hand. For PdM, this involves an in-depth analysis of the existing challenges in developing and deploying PdM applications. The motivation for this research is rooted in the need to enhance the efficiency and reliability of maintenance processes in industrial settings, particularly within the automotive sector. This stage sets the foundation for the research by establishing the context and justifying the need for an innovative solution (Hevner et al., 2004).

Define the Objectives for a Solution: Building upon the identified problem, the next step is to articulate clear and measurable objectives for the solution. This involves specifying the desired features and capabilities of the *Predictive Maintenance Framework*, such as scalability, interoperability, and real-time data processing. The objectives are derived from both the practical requirements of the industry and the theoretical constructs of PdM (Peffers et al., 2007).

Design and Development: At the heart of DSR is the design and development of the artifact, which, in this case, is a component repository tailored for PdM applications. This stage involves iterative cycles of prototyping, testing, and refinement to construct a solution that aligns with the predefined objectives. The design process is informed by a synthesis of existing knowledge and innovative approaches to CBSE (March and Smith, 1995).

Demonstration: The demonstration phase involves showcasing the artifact in operation within a controlled environment or real-world setting. For this research, the component repository is integrated into an automotive PdM application to illustrate its functionality and to provide a proof of concept. This stage is critical for validating the design choices and for illustrating the artifact's capability to address the identified problem (A. Hevner and Chatterjee, 2010).

Evaluation: A thorough evaluation follows the demonstration, where the artifact's performance is assessed against the objectives. This evaluation is multifaceted, encompassing both qualitative and quantitative measures, to ensure a comprehensive understanding of the artifact's impact and effectiveness. The evaluation phase is fundamental for providing evidence of the artifact's benefits and for guiding further refinements (Venable, Pries-Heje, and Baskerville, 2012).

Communication: The culmination of the DSR process is the communication of the findings. This involves documenting the research process, the developed artifact, and the results of the evaluation. The knowledge disseminated through this stage contributes to the academic literature and provides valuable insights for practitioners in the field of PdM (Gregor and A. R. Hevner, 2013).

The iterative nature of DSR allows for continuous learning and improvement, ensuring that the developed PdM component repository is not only theoretically sound but also practically viable and effective.

The integration of DSR within the automotive industry's PdM initiatives exemplifies the methodology's robustness and its vital role in fostering innovation at the intersection of technology and industry-specific challenges. This research leverages DSR as a strategic scaffold, guiding the systematic development of a PdM framework that is both innovative and practically viable. The iterative design and evaluation cycles intrinsic to DSR are particularly beneficial for the complex and evolving nature of automotive PdM, ensuring that each iteration refines the framework towards an optimal solution.

Iterative Design in Automotive Context: At the heart of applying DSR to automotive PdM is the iterative design and development of a component-based framework. Each cycle includes conceptualizing component designs tailored to automotive requirements, developing prototypes, integrating with existing PdM systems, and conducting preliminary evaluations. This approach ensures that the framework remains attuned to the specific needs of the automotive industry, including compliance with stringent safety and reliability standards (Rajkumar et al., 2010).

Machine Learning and AI Integration: A significant aspect of the DSR application in this context is the incorporation of ML and AI technologies. By iteratively designing and refining AI-driven components, the framework benefits from enhanced predictive capabilities, leading to more accurate maintenance forecasting and optimized resource allocation (J. Lee, Bagheri, and Kao, 2015).

Component-Based Software Engineering: The DSR methodology is particularly well-suited for advancing CBSE within the realm of PdM. Through iterative development, components can be systematically evaluated and improved, ensuring that they not only perform their intended functions but also seamlessly integrate with other system elements, thereby enhancing the overall robustness and modularity of the *Predictive Maintenance Framework* (Crnkovic, Stafford, et al., 2004).

Practical and Theoretical Advancements: The iterative cycles of design and evaluation in DSR serve to validate the framework's practicality within the automotive sector and contribute to theoretical knowledge in both CBSE and PdM. Each cycle yields insights that are instrumental in refining the framework and expanding the academic literature on the application of CBSE in PdM (Hevner et al., 2004).

In summary, the application of DSR in developing a PdM framework for the automotive industry is characterized by a strategic fusion of practical engineering and theoretical research. The iterative design and evaluation cycles are crucial in creating a robust, adaptable framework that not only addresses current industry challenges but is also poised to evolve with future technological advancements.

The integration of DSR within PdM research is vital in confronting the elaborate questions that underpin this thesis. The methodology's systematic approach to innovation, its focus on the creation of practical artifacts, and its dedication to iterative solution refinement is its core strengths. This research, through DSR, establishes a foundation for a PdM framework that is robust in theory and viable in practice.

The theoretical contributions of this study are rooted in the DSR framework's ability to allow a profound engagement with the academic literature on PdM, leading to a framework that is well-founded in scholarly research. The iterative cycles inherent in DSR ensure that each phase of design and evaluation enriches the theoretical understanding of the field, thereby extending the academic discourse of PdM.

On the practical front, the DSR methodology yields concrete artifacts that are specifically designed to meet the demands of the automotive industry. These artifacts exemplify the framework's real-world applicability and underscore the feasibility of the solutions it proposes. Such outcomes not only reflect the innovative spirit of this research but also its readiness for implementation within the industry.

Furthermore, the prescriptive nature of DSR spurs innovation in PdM, prompting the exploration of novel methodologies and the integration of emerging technologies. The framework that emerges from this research signifies a substantial advancement in applying CBSE to PdM, setting a precedent for future innovation in the sector.

In sum, the adoption of DSR is key in addressing the research challenges of PdM within the automotive industry. The developed framework is a clear indication of the potential that DSR holds in not just answering complex research questions but also in paving the way for the development of sophisticated PdM solutions.

3.2 Research Environment for Data Collection

Advancements in OTA connectivity catalyze the emergence of PdM for customer vehicles within the automotive industry. This new application domain for PdM enhances customer experience by proactively identifying maintenance needs, thereby reducing maintenance costs, improving spare part logistics, and increasing transparency in component behavior. Such improvements are vital for data-driven product design decisions, ultimately enhancing development efficiency (S. M. Lee, D. Lee, and Y. S. Kim, 2019; Donghwan Kim, S. Lee, and Daeyoung Kim, 2021).

In contrast to the manufacturing sector, where PdM data flow is often decentralized across various functional groups, the automotive sector benefits from a centralized data architecture. Customer vehicle data is aggregated through a single interface, managed by a dedicated organizational team. This data, transmitted from the Electronic Control Unit (ECU) of each vehicle via mobile networks, is crucial for enabling PdM services (Arena et al., 2021; Poor, Basl, and Zenisek, 2019;

Groba et al., 2007).

The centralization of this data interface is not merely a convenience but a necessity, driven by the substantial efforts required in data collection and preprocessing. Unlike domain knowledge, the statistical and mathematical expertise needed for data science in PdM is more closely aligned with data engineering and architecture, advocating for a centralized model development approach (C. Chen et al., 2021; Bekar, Nyqvist, and Skoogh, 2020).

The automotive industry is undergoing a significant transformation with the adoption of centralized Electrical/Electronical (E/E) architectures, driven by the increasing complexity of software in modern vehicles (Bandur et al., 2021). Centralized E/E architectures offer numerous advantages over traditional decentralized systems, such as improved cost-efficiency, lower latency, enhanced flexibility, and increased reliability (Kanajan et al., 2006). These centralized systems are particularly advantageous for PdM applications, where data from various sensors and ECU need to be aggregated and analyzed to anticipate maintenance needs.

In the context of automotive OEMs, a centralized data architecture is not just a strategic choice but a necessity. The data from customer vehicles, collected by ECUs, is transmitted via mobile networks to a single backend server. This consolidation is managed by a dedicated organizational team, which enables the development of PdM services by other teams within the company (Ren, Yingnan Liu, and S. Zhang, 2022). Such a centralized approach simplifies the data flow, which in manufacturing contexts is often diverse and decentralized, leading to a fragmented responsibility across different functional groups.

The integration of sensors in automotive PdM is fraught with challenges, particularly due to the cost implications of mass production and the need for a compelling business case to justify their inclusion in vehicles. Sensors that are critical for PdM must provide significant value to outweigh these costs, often leading to a reliance on sensors justified by other vehicle functions. This constraint demands extensive data preparation and feature engineering to derive meaningful insights for PdM (Teepe and Görnig, 2003).

Moreover, the transmission of data from these sensors presents its own set of challenges. The sensitivity of customer data, governed by stringent privacy laws, requires careful handling to ensure compliance and maintain customer trust. The architecture must, therefore, be designed to handle the secure and efficient transmission of potentially large volumes of data, while also considering the limitations of global network infrastructure and the costs associated with transmission hardware (Pathan, Hyung-Woo Lee, and Choong Seon Hong, 2006).

The need for centralization in PdM model development within the automotive industry is underscored by the elaborate challenges associated with data collection and preprocessing. Centralized data management systems are vital for the efficient handling of the vast amounts of data generated by customer vehicles, which is essential for the effective implementation of PdM (Razali et al., 2020). The consolidation of data from various sensors and ECUs into a single backend server, as managed by a dedicated organizational team, not only reorganizes the process but also enhances the predictive capabilities by leveraging big data analytics. This centralized approach is particularly beneficial in the context of automotive OEMs, where it ensures the generalizability of PdM models across different entities within a major automotive group, thereby paving the way for a unified strategy for maintenance and service improvements (Kong, 2022).

The synergy between domain knowledge experts and data scientists is key in the realm of PdM. Domain experts bring a nuanced understanding of the specific maintenance requirements and operational intricacies of automotive components. In contrast, data experts use the analytical expertise to interpret complex datasets and extract actionable insights. This interdisciplinary collaboration is essential for developing robust PdM models that are both accurate and applicable in real-world scenarios (Löffler, Von Der Linden, and Schneider, 2016; Park et al., 2021; Y.-a. Kang and Stasko, 2012; Chilana, Wobbrock, and Ko, 2010; Barley, Treem, and Leonardi, 2020; Roux et al., 2006; Olivier and Verschoof-van Der Vaart, 2021; Chande and Tokekar, 1998; Moser, 2017; Jamrozik and Gentner, 2020; A. A. Mitchell, Russo, and Wittink, 1991; Demetriadis et al., 2011; Sambasivan and Veeraraghavan, 2022; Will, Mcquaig, and Hardaway, 1994; O'Keefe, 1985; Mueller and Dyerson, 1999).

Studies show that domain-specific knowledge can enhance monitoring performance, although it may also lead to overconfidence in certain situations (Löffler, Von Der Linden, and Schneider, 2016). Tools like *Ziva* accelerate the sharing of domain knowledge, which helps data scientists learn essential information about the domain, offering scalability of information and lowering the burden on domain experts (Park et al., 2021). Moreover, case studies of prolonged system use by domain analysts working with their own data can inform the design implications for future systems, ensuring that visual analytics systems truly help expert users accomplish their goals (Y.-a. Kang and Stasko, 2012).

The literature also suggests that partnerships with domain experts lead to effective results, especially when domain experts are willing to be an integral part of the usability team (Chilana, Wobbrock, and Ko, 2010). Furthermore, the cultivation of process experts can support coordination among domain experts by focusing on the performance, production, and value of process expertise (Barley, Treem, and Leonardi, 2020). The concept of "co-production" of knowledge through collaborative learning between "experts" and "users" is proposed as a more suitable approach to building knowledge systems for sustainable management, as opposed to unidirectional knowledge transfer (Roux et al., 2006).

Cross-domain collaboration, where expert knowledge from different research fields is used, shows to create more reliable detectors and predictive models (Olivier and Verschoof-van Der Vaart, 2021). Additionally, expert-based maintenance improves the performance of systems, highlighting the effectiveness of expert systems in maintenance (Chande and Tokekar, 1998). The recognition of experts' status through individualized and public feedback is found to increase their contribution to knowledge sharing (Moser, 2017).

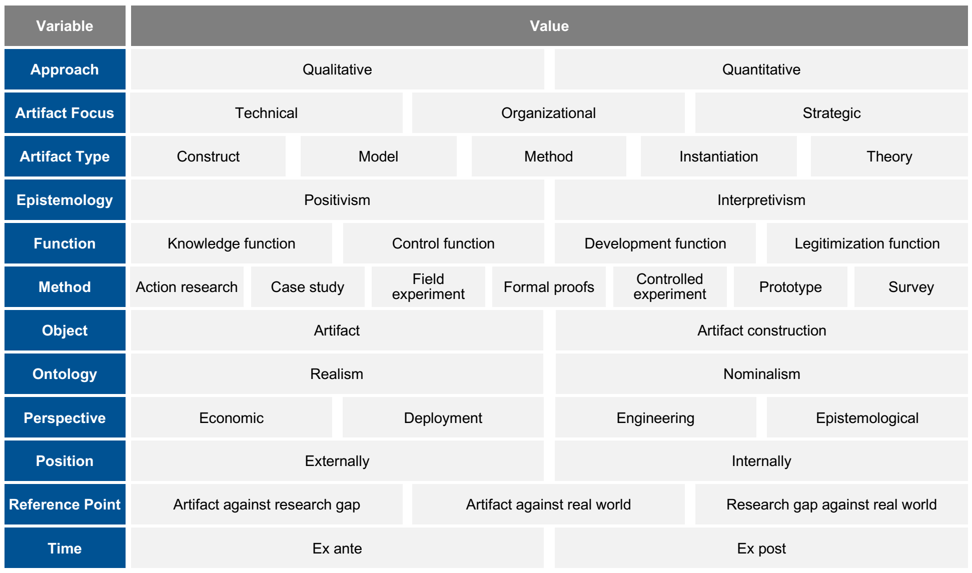
To sum up, the collaboration between domain knowledge experts and data experts is a dynamic and iterative process that benefits from structured communication, shared understanding, and mutual respect. The development of a structured framework within this research aims to formalize and enhance these collaborative dynamics, ultimately leading to more effective and generalizable PdM models.

3.3 Artifact Analysis and Directives

In this work, the classification of artifacts, central to any IS research, adheres to the methodologies portrayed by Cleven, Gubler, and Hüner (2009). This methodology underscores the vital role of the artifact in IS by integrating both behavioral and DSR paradigms, which guide the empirical and design-oriented scientific inquiries respectively. The selection of the artifact is primarily influenced by the research objectives, articulated through research questions or hypotheses to be tested, and is also contingent upon the resources available to the researcher. Notably, in DSR, these resources significantly impact the design decisions of the research structure.

Following the framework by Ritchey (2006), the artifact is first characterized using various variables from the morphological field. This process begins with a predominantly qualitative evaluation of the artifact, supported by quantitative elements to ensure a comprehensive understanding of the complex artifact structure and its scientific queries (W. Chen and Hirschheim, 2004). Furthermore, the epistemological foundation of the artifact, which navigates the dependency of scientific insights on their interpretation and assessment by the researcher, is defined as interpretivism following the guidelines by Siau and Rossi (2011). This approach is particularly important when the research methodology incorporates elements of Action Research, indicating no distinct separation between the researcher and the research subject.

The morphological field developed by Cleven, Gubler, and Hüner (2009) provides a structured framework for classifying and evaluating DSR artifacts. This systematic approach categorizes various dimensions and characteristics essential for a comprehensive assessment of artifacts within IS research. The detailed representation of this morphological field, illustrating its complex interrelationships and categorical distinctions, is displayed in Figure 12. This visualization aids in understanding the elaborate structure and utility of the morphological field in guiding the evaluation and classification of DSR artifacts across different dimensions and criteria.



Variable	Value						
Approach	Qualitative	Quantitative					
Artifact Focus	Technical	Organizational	Strategic				
Artifact Type	Construct	Model	Method	Instantiation	Theory		
Epistemology	Positivism	Interpretivism					
Function	Knowledge function	Control function	Development function	Legitimization function			
Method	Action research	Case study	Field experiment	Formal proofs	Controlled experiment	Prototype	Survey
Object	Artifact	Artifact construction					
Ontology	Realism	Nominalism					
Perspective	Economic	Deployment	Engineering	Epistemological			
Position	Externally	Internally					
Reference Point	Artifact against research gap	Artifact against real world	Research gap against real world				
Time	Ex ante	Ex post					

Figure 12: Morphological field for DSR artifact classification with various dimensions (values) per category (variable) by Cleven, Gubler, and Hüner (2009).

In the classification and evaluation of DSR artifacts, the **Approach** category critically evaluates the characteristics of the artifact using qualitative, quantitative, or mixed methods. Qualitative approaches focus on value-based assessments, emphasizing the description and deeper understanding behind the observable factors, often transforming qualitative data into numerical form through codification for analytical purposes. Quantitative approaches, on the other hand, rely on numerical data to provide statistical analysis, offering precise measurement and comparison capabilities. Mixed methods combine both qualitative and quantitative techniques to harness the strengths of each, providing a more robust analysis that leverages detailed, context-rich insights from qualitative data alongside the empirical rigidity of quantitative data. This multifaceted evaluation framework allows for a comprehensive assessment of the artifact, catering to the varied nature of research questions and hypotheses in DSR.

The **Artifact Focus** category identifies the intended use context of DSR artifacts, categorizing them into technical, organizational, or strategic types based on their application environment. Technical artifacts, such as routing algorithms and hardware designs, are primarily concerned with the operational and infrastructural aspects of systems, enhancing technical efficiency and capability. Organizational artifacts include tools like process models and methods for organizational redesign, aimed at improving workflows, communication, and procedural structures within a company. Strategic artifacts, such as decision support systems and roadmap development methods, are designed to align with and support long-term business goals, allowing strategic decision-making and future planning. This focus differentiation is fundamental as it guides the development and evaluation processes, ensuring that the artifacts are optimally aligned with their respective application domains and contribute effectively to achieving intended outcomes in DSR.

The **Artifact Type** category encompasses a diverse range of artifact forms within DSR, each serving distinct functions. **Constructs** are foundational elements that provide the vocabulary and syntax for defining and understanding problems and solutions within a domain, such as classification systems and ontologies. **Models** articulate relationships between constructs and abstract aspects of reality to simplify and clarify complex systems, often represented through meta-models or reference models. **Methods** offer systematic approaches comprising algorithms and procedures tailored to solve specific classes of problems, such as techniques for business process modeling or software development. **Instantiations** are concrete implementations of constructs, models, or methods, demonstrating the feasibility of ideas and allowing for practical testing in real-world conditions. Finally, **Theories** in DSR articulate and predict cause-and-effect relationships, serving not only as a theoretical backbone for projects but also as outcomes that provide new insights into the field. This classification into constructs, models, methods, instantiations, and theories facilitates a structured approach to artifact creation and evaluation, ensuring that each artifact type is appropriately developed and assessed based on its intended function and impact in DSR.

The **Epistemology** category in DSR focuses on the theoretical underpinnings that guide the interpretation and validation of research findings, directly reflecting the influence of the researcher's characteristics on the evaluation process. **Positivism** presumes that knowledge derived from the scientific method is seen as objective, and that true knowledge is obtained when it is free from the researcher's biases, focusing strictly on observable phenomena and measurable facts.

This approach supports the notion that evaluation outcomes should be replicable and consistent, regardless of who conducts the research. In contrast, **Interpretivism** suggests that knowledge is subjective and constructed, emphasizing the importance of understanding the phenomena through the context in which they are interrelated. Here, the researcher's perspectives and interpretations play a crucial role in shaping the findings, as it is believed that each researcher brings a unique set of experiences and biases that can influence the outcomes. These epistemological stances impact how artifacts are evaluated within DSR, determining whether the emphasis is placed on objective measurements and replicability (positivism) or contextual understanding and the richness of subjective insights (interpretivism).

The **Function** of evaluation within DSR is categorized into knowledge, controlling, development, and legitimization, each contributing uniquely to the comprehensive assessment of artifacts. The **Knowledge Function** is fundamental, aiming to generate insights that inform decision-making and strategy. It involves the systematic collection and analysis of data to derive actionable intelligence that can substantiate managerial and technical decisions. The **Controlling Function** uses evaluation to verify whether the artifact meets predefined criteria such as efficiency, effectiveness, and user acceptance, serving as a regulatory mechanism to ensure that the artifact achieves its intended goals. The **Development Function** is oriented towards iterative improvement, utilizing feedback from the evaluation process to refine and enhance the artifact. This function supports a cycle of continuous development and adaptation, which is key in responding to evolving user needs and environmental changes. Lastly, the **Legitimization Function** seeks to justify the existence and use of the artifact by demonstrating its value and usefulness through structured evaluation. This function is particularly important in securing stakeholder support and funding, as it provides empirical evidence of the artifact's benefits and alignment with organizational or societal objectives.

The **Methods** of evaluation in DSR encompass a wide range of approaches, each tailored to suit specific research contexts and objectives. **Action Research** is a participative and iterative method where the researcher works closely with participants to address a real-world problem, supporting change and generating practical knowledge. This method is especially suitable for dynamic environments where ongoing modifications and continuous feedback are essential. **Case Studies** are employed to conduct in-depth investigations into the specific application or functioning of an artifact within its real-life context, providing comprehensive insights into complex phenomena. **Field Experiments** involve manipulating one or more variables in a natural setting to observe the effects on some outcomes, making it possible to study the causal impact of an artifact under controlled, yet realistic conditions. **Controlled Experiments** are more structured and are conducted in settings where variables can be controlled thoroughly to isolate the effects of the artifact on specific outcomes, often used in laboratory environments to ensure reliability and replicability of results. The use of **Prototyping** allows for the practical demonstration and testing of an artifact's design and functionality in a working state, providing immediate feedback on its performance and user interactions. These methods, from action research to surveys, provide diverse tools for researchers to validate the usefulness and suitability of DSR artifacts, each contributing uniquely to the depth and breadth of the evaluation process. Lastly, **Surveys** are utilized to collect quantitative or qualitative data from a predefined group of respondents, providing insights into perceptions, effectiveness, and user satisfaction related to the artifact. This method is particularly valuable for gauging broad user feedback and for statistical analysis of user interactions with the artifact.

The **Object** of evaluation in DSR can be broadly categorized into two primary focuses: the artifact itself and the process by which the artifact is constructed. Evaluating the **Artifact Itself** involves assessing the tangible outputs of the DSR process. This evaluation examines the functionality, effectiveness, and usability of the artifact, ensuring it meets the specified requirements and performs optimally within its intended environment. Such evaluations often involve user testing, performance benchmarking, and real-world application scenarios to gauge the artifact's practical value and impact. On the other hand, evaluating the **Construction Process** of the artifact focuses on the methodologies, tools, and procedures employed during the artifact's development. This form of evaluation looks at the efficiency, adaptability, and rigidity of the development processes, assessing aspects such as methodological soundness, adherence to design principles, and the overall workflow efficacy. By evaluating the construction process, researchers can identify areas for process improvement, increase the reproducibility of the artifact, and ensure that the design process itself adheres to high standards of quality and innovation. Both evaluation objects are essential; the former ensures the artifact's direct applicability and effectiveness, while the latter guarantees that the foundational processes are robust, transparent, and capable of producing reliable and high-quality results.

The **Ontology** category in DSR addresses the philosophical study of the nature of being, existence, or reality as it relates to artifacts. This exploration focuses particularly on the assumptions about the reality of the artifacts and their attributes. **Realism** assumes that the existence of artifacts, as well as their properties, are objective and independent of human thought or perception. From this viewpoint, artifacts are considered to have an existence and a set of characteristics that are tangible and detectable, regardless of whether or not they are observed. This perspective is fundamental for DSR as it supports the idea that artifacts can be empirically tested and validated, providing a stable framework for scientific inquiry. On the other hand, **Nominalism** challenges this by suggesting that the properties and classifications of artifacts are products of human conception and linguistic practices rather than inherently existing in the artifacts themselves. According to nominalism, the characteristics of artifacts are not universal but are instead constructed through social interactions and agreements among communities. This view impacts DSR by emphasizing the contextual and negotiated nature of how artifacts are understood and valued, which can vary significantly between different groups or cultures. Understanding these ontological positions helps in framing the evaluation and development of artifacts within DSR, acknowledging how perceptions of reality can influence design choices and the interpretation of results.

The **Perspective** from which an artifact is evaluated in DSR encompasses a variety of viewpoints, each providing unique insights into the artifact's value and performance. The **Economic Perspective** focuses on the cost-effectiveness, Return on Investment (ROI), and overall economic impact of the artifact. This perspective assesses whether the artifact provides a viable financial benefit compared to its cost, considering not only direct expenses but also long-term savings and profitability enhancements. The **Deployment Perspective** examines how well the artifact integrates into existing systems and workflows, its ease of implementation, and its acceptance by end-users. It emphasizes the practical aspects of deploying the artifact in real-world settings, including user training needs and the level of disruption to current processes. The **Engineering Perspective** investigates the technical design and architectural robustness of the artifact, analyzing the quality of engineering practices used in its creation. This includes assessments of scalability, maintainability, and compliance with technical standards. Lastly, the **Epistemological Perspective** explores the knowledge foundations of the artifact, questioning how knowledge is created and validated within the artifact's domain. It challenges the assumptions and methodologies underlying the artifact's design and seeks to understand the theoretical basis from which the artifact is developed. Each of these perspectives offers critical lenses through which the artifact's utility, feasibility, and theoretical grounding can be precisely evaluated, ensuring a comprehensive assessment across multiple dimensions of interest.

The **Position** of evaluation in DSR refers to whether the assessment is conducted internally or externally, each offering different benefits and insights. **Internal Evaluation** is performed by individuals who are directly involved in the artifact's development. This allows for a deep understanding of the artifact's design and intentions, enabling detailed and nuanced feedback. Internal evaluators can leverage their intimate knowledge of the artifact to conduct thorough testing and to identify subtle issues or areas for improvement that external evaluators might overlook. However, this type of evaluation may be subject to biases since the evaluators have a vested interest in the artifact's success. On the other hand, **External Evaluation** is carried out by individuals who are not part of the development team. This position promotes objectivity and can provide a fresh perspective on the artifact's usability and effectiveness in real-world settings. External evaluators are likely to provide impartial feedback, which can be key for identifying usability problems or other issues that internal evaluators might miss due to their familiarity with the artifact. Furthermore, external evaluations can help in assessing the artifact's generalizability and its performance across different contexts and user groups. Both positions are vital for a balanced evaluation strategy, with internal evaluations offering depth and insider insight, while external evaluations provide breadth and unbiased critique.

The **Reference Point** in the evaluation of DSR artifacts serves as a critical benchmark for assessing the relevance and usefulness of the artifact. This involves comparing the artifact against the research gap, real-world applications, or both, to validate its practical and theoretical contributions. Evaluating the **Artifact against the Research Gap** involves examining whether the artifact successfully addresses the identified deficiencies or needs within the existing body of knowledge. This assessment ensures that the artifact contributes novel insights or solutions that are not just incremental improvements but significant advancements in the field. The focus here is on the artifact's ability to fill theoretical voids or enhance methodological approaches.

Evaluating the **Artifact against Real-World Applications** tests the artifact in practical scenarios to ensure that it not only functions theoretically but also performs effectively in real-world settings. This evaluation helps in verifying the artifact's utility, scalability, and robustness outside the controlled research environment. It provides a measure of how well the artifact translates abstract concepts into tangible outcomes that can solve actual problems faced by organizations or industries.

Lastly, some evaluations may involve a **Dual Reference Point**, where the artifact is assessed both in terms of its theoretical contribution (against the research gap) and its practical applicability (in real-world applications). This dual approach ensures a comprehensive evaluation, demonstrating the artifact's relevance and usefulness comprehensively across both academic and practical dimensions. Such a thorough assessment helps in substantiating the artifact's overall value, promoting broader acceptance and adoption.

The **Time** of evaluation in DSR plays a crucial role in shaping the insights and outcomes of the assessment process. This temporal aspect of evaluation is categorized into **ex ante** and **ex post**, each offering unique perspectives on the artifact's development and implementation stages. **Ex ante evaluation** occurs before the artifact is fully implemented, serving as a predictive analysis to foresee the potential impacts and effectiveness of the artifact. This type of evaluation is primarily concerned with theoretical validation and feasibility assessments. It aims to preemptively identify possible challenges, limitations, or the need for additional refinements before the artifact is deployed. Ex ante evaluations help in ensuring that the artifact is adequately prepared to meet the intended goals and is likely to succeed in its operational environment.

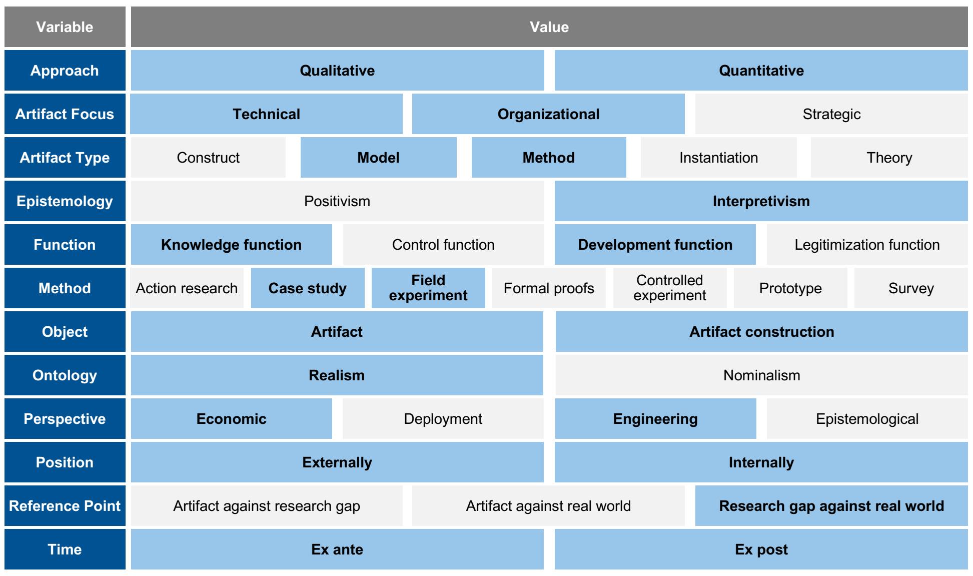
On the other hand, **ex post evaluation** is conducted after the artifact's implementation. This evaluation focuses on the actual outcomes and real-world impacts of the artifact, assessing its effectiveness, user adoption, and overall performance. Ex post evaluations provide empirical data that can validate the artifact's utility and inform further developments. This retrospective assessment is vital for understanding the artifact's practical implications and for drawing lessons that can be applied to future projects or ongoing improvements.

Both ex ante and ex post evaluations are essential for a comprehensive assessment strategy in DSR, allowing researchers and practitioners to critically analyze and refine the artifact at different stages of its lifecycle. Ex ante evaluations guide the initial development and deployment strategies, while ex post evaluations offer feedback based on actual usage and results, ensuring the artifact achieves its intended purpose and adapts to user needs and environmental changes over time.

These evaluations not only guide initial development and deployment strategies but also provide essential feedback after implementation, ensuring that the artifacts meet their intended purposes and adapt effectively to user needs and environmental changes. The stringent application of these evaluation methods, alongside the other categorized dimensions discussed, such as Approach, Artifact Focus, Artifact Type, and others, is crucial for substantiating the utility and impact of artifacts within their respective fields. As this work progresses, the following sections will present the detailed discussion of the five artifacts developed, each evaluated using the outlined systematic approach. This will showcase how these artifacts address specific research gaps and practical challenges, highlighting their contributions and the potential for future enhancements.

3.3.1 Artifact Analysis of the Predictive Maintenance Framework

The first artifact, a **Predictive Maintenance Framework**, is designed using a structured literature analysis combined with a two-layer topic modeling NLP approach, applying it to scientific literature tagged with `predictive maintenance`. This sophisticated approach allows the artifact to modularize PdM into four distinct phases, each consisting of two to three components, answering the research question on the feasibility of classifying PdM applications within a systematic framework. The artifact's evaluation includes traditional literature research and field observation in a real-world environment relevant to this work. Figure 13 shows the artifact classification according to the presented morphological field.



Variable	Value						
Approach	**Qualitative**	**Quantitative**					
Artifact Focus	**Technical**	**Organizational**	Strategic				
Artifact Type	Construct	**Model**	**Method**	Instantiation	Theory		
Epistemology	Positivism	**Interpretivism**					
Function	**Knowledge function**	Control function	**Development function**	Legitimization function			
Method	Action research	**Case study**	**Field experiment**	Formal proofs	Controlled experiment	Prototype	Survey
Object	**Artifact**	**Artifact construction**					
Ontology	**Realism**	Nominalism					
Perspective	**Economic**	Deployment	**Engineering**	Epistemological			
Position	**Externally**	**Internally**					
Reference Point	Artifact against research gap	Artifact against real world	**Research gap against real world**				
Time	**Ex ante**	**Ex post**					

Figure 13: Classification of the first artifact, *Predictive Maintenance Framework*, according to the morphological field for DSR artifact classification (own figure).

- **Approach:** Utilizing a *Mixed Methods* strategy, the framework integrates *qualitative insights* from thematic categorization with *quantitative analyses* from topic modeling. This hybrid approach enhances the depth and validity of the framework by leveraging both narrative context and empirical data.
- **Artifact Focus:** The framework impacts both *Technical* and *Organizational* levels within an enterprise. Technically, it refines maintenance protocols and procedures; organizationally, it influences maintenance strategy and policy.
- **Artifact Type:** Classified as both a *Model* and a *Method*, the framework provides a

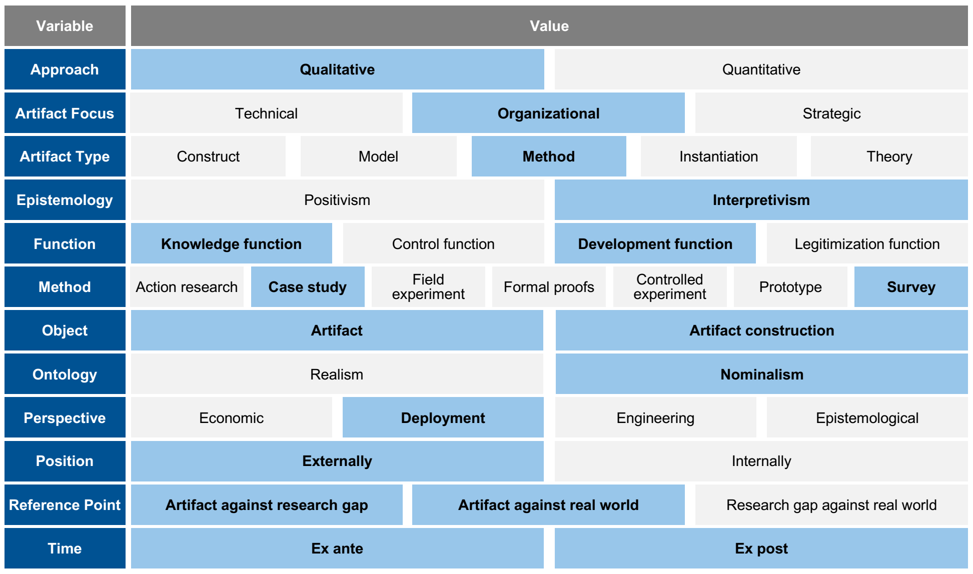
structural model of PdM phases and a methodological approach to implement these phases effectively within organizational practices.

- **Epistemology:** The development leans towards *Interpretivism*, acknowledging that the modularization of PdM is influenced by the subjective interpretation of literature and observed practices, thus embracing the complexity and variability of maintenance needs across different industries.
- **Function:** It primarily addresses the *Knowledge Function* by systematizing existing concepts and theories into a practical framework, and the *Development Function* by proposing a structured approach to enhance maintenance operations.
- **Methods:** The evaluation employs both *Case Study* for theoretical grounding and *Field Experiment* for practical validation, ensuring the framework's applicability and robustness in real-world settings.
- **Object:** The evaluation focuses on the *Artifact Itself*, assessing the framework's capability to improve maintenance processes, and on the *Construction Process*, ensuring the methodological rigidity and comprehensiveness of the research approach.
- **Ontology:** Adheres to *Realism*, under the assumption that the structured phases of PdM exist objectively and can be empirically tested and validated within the operational environment.
- **Perspective:** Evaluated from both *Engineering* and *Economic Perspectives*, considering the framework's operational efficiency and cost-effectiveness, respectively.
- **Position:** Evaluation is conducted both *Internally* by the research team and *Externally* by industry practitioners, offering a comprehensive view of the framework's effectiveness and adaptability.
- **Reference Point:** Assessed against the *Research Gap* it aims to fill and its performance in *Real-World Applications*, demonstrating both theoretical relevance and practical utility.
- **Time:** Includes both *Ex Ante* evaluations to predict potential impacts before full-scale implementation, and *Ex Post* evaluations to assess actual outcomes post-deployment.

The *Predictive Maintenance Framework*, developed through a structured literature analysis and a two-layer topic modeling NLP approach, represents a significant advancement in the modularization of PdM. This framework systematically categorizes PdM into four distinct phases, each comprising several components, thus providing a comprehensive model that addresses the initial research question of classifiability within a systematic framework. Evaluated using mixed methods, the framework combines qualitative insights from literature with quantitative data from field observations, enhancing its empirical robustness and practical applicability. It spans both technical and organizational aspects of PdM, offering strategies that enhance maintenance operations across various industry sectors. The framework's development, grounded in interpretivism, reflects a deep understanding of the complexities involved in PdM. Furthermore, its evaluation across multiple dimensions—including technical efficiency and economic viability—ensures its relevance and usefulness in real-world settings. The methodological rigidity and collaborative evaluation approach, encompassing both internal insights and external validations, position this artifact as a foundational tool in the evolving field of PdM, promising significant contributions to both academic research and industrial practice.

3.3.2 Artifact Analysis of the Use Case Description Methodology

The second artifact, a **Use Case Description Methodology**, is developed through expert interviews, literature research, and analysis of real-world PdM models. This methodology employs a canvas-based approach to plan and manage the development of PdM use cases. It systematically integrates input prerequisites with output requirements and Dp, and provides a comprehensive overview of the stakeholders involved. This artifact is evaluated through its application to existing use cases and feedback from domain experts. Figure 14 illustrates the classification of this artifact according to the established morphological field.



Variable	Value						
Approach	**Qualitative**	Quantitative					
Artifact Focus	Technical	**Organizational**	Strategic				
Artifact Type	Construct	Model	**Method**	Instantiation	Theory		
Epistemology	Positivism	**Interpretivism**					
Function	**Knowledge function**	Control function	**Development function**	Legitimization function			
Method	Action research	**Case study**	Field experiment	Formal proofs	Controlled experiment	Prototype	**Survey**
Object	**Artifact**	**Artifact construction**					
Ontology	Realism	**Nominalism**					
Perspective	Economic	**Deployment**	Engineering	Epistemological			
Position	**Externally**	Internally					
Reference Point	**Artifact against research gap**	**Artifact against real world**	Research gap against real world				
Time	**Ex ante**	**Ex post**					

Figure 14: Classification of the second artifact, *Use Case Description Methodology*, according to the morphological field for DSR artifact classification (own figure).

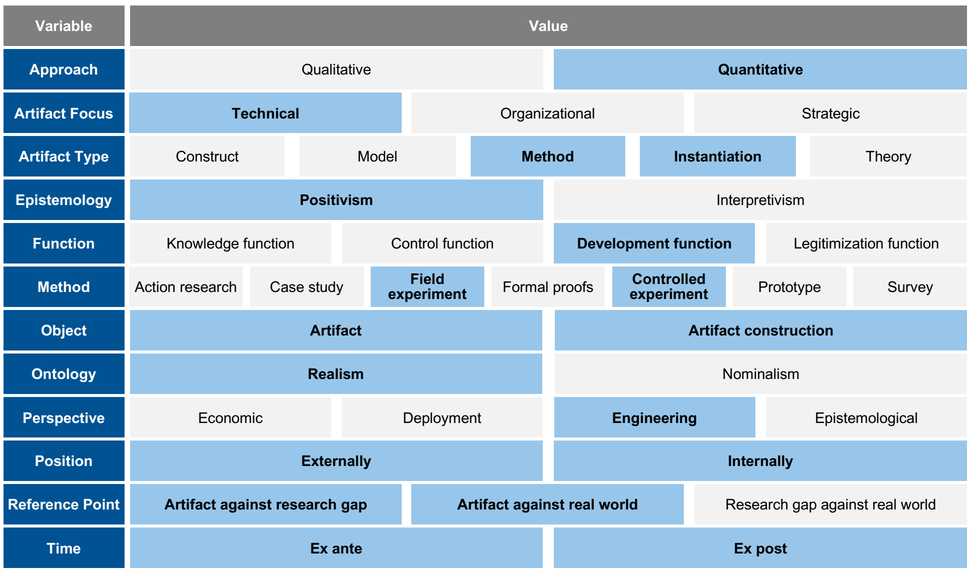
- **Approach:** This methodology employs a *Qualitative* approach, heavily relying on expert insights and detailed analysis of literature and existing models to derive a structured method for PdM use case development.

- **Artifact Focus:** Primarily *Organizational*, as it aims to enhance the planning and management of PdM projects within organizations by clarifying roles and expectations through a detailed use case canvas.
- **Artifact Type:** This is a *Method*, providing a structured process for stakeholders to follow when developing new PdM use cases.
- **Epistemology:** Reflecting an *Interpretivist* approach, the methodology is developed from the subjective interpretations of experts and tailored to the specific context of PdM in which it is applied.
- **Function:** Addresses the *Knowledge* and *Development* functions by organizing and clarifying the use case development process, thus facilitating better understanding and implementation of PdM projects.
- **Methods:** Evaluated using *Case Study* and *Expert Feedback*, which provide insights into the practical applicability and effectiveness of the methodology in real-world settings.
- **Object:** Focuses on the *Artifact Itself*—the methodology—as well as on the *Construction Process*, evaluating both the end product and the methods used in its creation.
- **Ontology:** Adopts a *Nominalist* perspective, suggesting that the structure and components of use cases are defined through social agreements among stakeholders rather than existing as independent realities.
- **Perspective:** Evaluated from an *Organizational Perspective*, considering how the methodology affects the structure and efficiency of project management within organizations.
- **Position:** The evaluation is *Externally* performed by domain experts who are not involved in the development of the methodology, ensuring an unbiased assessment.
- **Reference Point:** Assessed against both the *Research Gap*, addressing the need for a structured use case development method in PdM, and its effectiveness in *Real-World Applications*.
- **Time:** Includes both *Ex Ante* evaluations to assess potential utility before widespread implementation, and *Ex Post* evaluations to observe its effectiveness in actual use.

The Use Case Description Methodology stands out as a critical tool in the PdM landscape, systematically structuring the development of use cases to ensure comprehensive planning and effective management. Its development, grounded in qualitative insights and interpretivism, underscores its adaptability and relevance in various organizational contexts, offering a valuable asset for both academic and practical applications in PdM.

3.3.3 Artifact Analysis of the Component Repository Structure

The third artifact, **Component Repository and Interface Standardization**, builds on the foundations of the previous PdM Framework and Use Case Description Methodology. This artifact focuses on creating a modular repository for the development of PdM models, emphasizing the importance of individual functional components. It employs CBSE principles to enhance the structure and reusability of these components. This approach not only integrates the insights from previous artifacts but also introduces a set of component design rules based on a priori reusability. The validation of this artifact involves its application to real-world use cases and the quantitative measurement of efficiency improvements attributable to the reusability of components. Figure 15 shows the classification of this artifact according to the morphological field.



Variable	Value						
Approach	Qualitative	**Quantitative**					
Artifact Focus	**Technical**	Organizational	Strategic				
Artifact Type	Construct	Model	**Method**	**Instantiation**	Theory		
Epistemology	**Positivism**	Interpretivism					
Function	Knowledge function	Control function	**Development function**	Legitimization function			
Method	Action research	Case study	**Field experiment**	Formal proofs	**Controlled experiment**	Prototype	Survey
Object	**Artifact**	**Artifact construction**					
Ontology	**Realism**	Nominalism					
Perspective	Economic	Deployment	**Engineering**	Epistemological			
Position	**Externally**	**Internally**					
Reference Point	**Artifact against research gap**	**Artifact against real world**	Research gap against real world				
Time	**Ex ante**	**Ex post**					

Figure 15: Classification of the third artifact, *Component Repository*, according to the morphological field for DSR artifact classification (own figure).

- **Approach:** The methodology adopts a *Quantitative* approach by measuring the efficiency gains from reusability, supplemented by qualitative insights from its integration with earlier artifacts.
- **Artifact Focus:** It is primarily *Technical*, targeting the enhancement of software engineering practices within PdM model development.
- **Artifact Type:** This artifact serves as both a *Method* and an *Instantiation*, offering a structured process for component development and a tangible repository that exemplifies

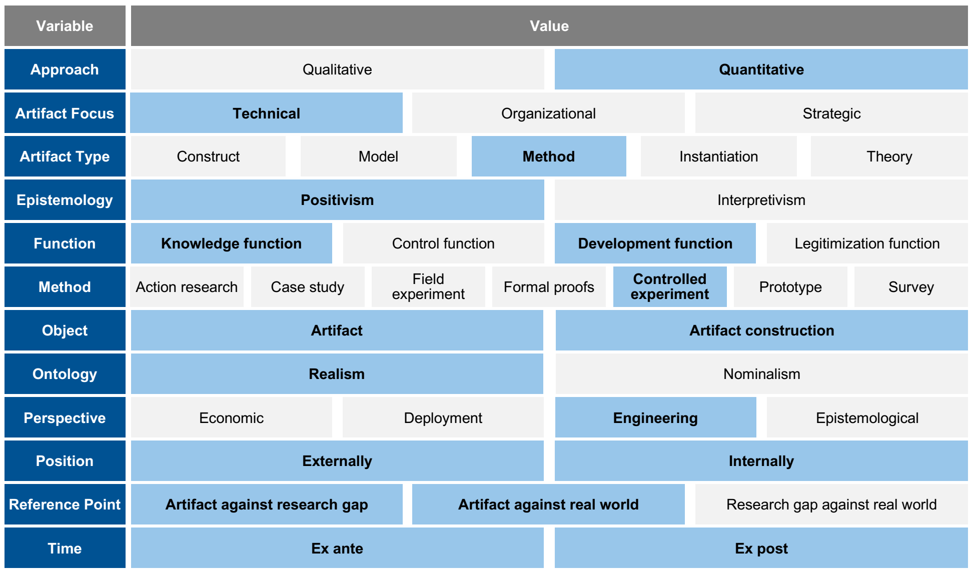
this process.

- **Epistemology:** The artifact follows a *Positivist* approach, aiming to provide objective, measurable outcomes from standardized practices and reusable components.
- **Function:** It significantly contributes to the *Development* function by standardizing component creation and usage, and the *Knowledge* function by documenting best practices for reusability.
- **Methods:** Evaluated using a combination of *Controlled Experiments* and *Field Experiments*, these methods confirm the practical efficacy and scalability of the repository in various operational settings.
- **Object:** The evaluation centers on the *Artifact Itself*—the repository—and the *Construction Process* that informs the creation of the standardization guidelines.
- **Ontology:** Grounded in *Realism*, it asserts that the components and their interfaces have objective characteristics that can be standardized and reused across different PdM systems.
- **Perspective:** Evaluated from an *Engineering Perspective*, it focuses on the technical soundness and applicability of the component designs and interfaces.
- **Position:** The artifact is evaluated *Internally* by the development team and *Externally* by third-party users to ensure its adaptability and effectiveness.
- **Reference Point:** It is assessed against the established *Research Gap* in component reusability and its operational performance in *Real-World Applications*.
- **Time:** Involves *Ex Ante* evaluations to forecast potential functionality and *Ex Post* evaluations to measure actual performance enhancements.

The Component Repository and Interface Standardization artifact is a fundamental development in the PdM field, promoting structured, efficient, and reusable component-based approaches. Its thorough evaluation, grounded in quantitative and qualitative methods, showcases significant advancements in the modularization and standardization of PdM models, offering substantial benefits for both theoretical research and practical applications.

3.3.4 Artifact Analysis of the Component Creation Workflow

The fourth artifact, the **Component Creation Workflow**, builds upon the structure established by the Component Repository and Interface Standardization. This workflow is designed to ensure efficient reusability by iteratively processing through three layers: model, module, and component. It provides developers with tools to describe the current object, search for existing objects, assess them for full or partial applicability, refine object descriptions, and create new objects only when necessary. The efficacy of this workflow is evaluated through a DES model, complemented by sensitivity analysis. The sensitivity analysis demonstrates parameter Dp and determines under which conditions this workflow outperforms traditional PdM model development methods. Figure 16 illustrates the classification of this artifact according to the morphological field.



Variable	Value						
Approach	Qualitative	**Quantitative**					
Artifact Focus	**Technical**	Organizational	Strategic				
Artifact Type	Construct	Model	**Method**	Instantiation	Theory		
Epistemology	**Positivism**	Interpretivism					
Function	**Knowledge function**	Control function	**Development function**	Legitimization function			
Method	Action research	Case study	Field experiment	Formal proofs	**Controlled experiment**	Prototype	Survey
Object	**Artifact**	**Artifact construction**					
Ontology	**Realism**	Nominalism					
Perspective	Economic	Deployment	**Engineering**	Epistemological			
Position	**Externally**	**Internally**					
Reference Point	**Artifact against research gap**	**Artifact against real world**	Research gap against real world				
Time	**Ex ante**	**Ex post**					

Figure 16: Classification of the fourth artifact, *Component Creation Workflow*, according to the morphological field for DSR artifact classification (own figure).

- **Approach:** This workflow utilizes a *Quantitative* approach through the use of DES and sensitivity analysis to measure efficiency gains and parameter impacts accurately.
- **Artifact Focus:** It primarily focuses on the *Technical* aspects of PdM development, optimizing the process of component creation within a structured repository environment.
- **Artifact Type:** Classified as a *Method*, this workflow provides a systematic procedure for managing component lifecycle within PdM projects.
- **Epistemology:** The development of this workflow is grounded in *Positivism*, relying on empirical data and structured simulations to validate its effectiveness.
- **Function:** It serves the *Development* function by enhancing the efficiency of component creation and the *Knowledge* function by generating empirical data on process optimizations.
- **Methods:** Evaluated using *Controlled Experiments* in the form of simulations, providing precise, controlled measurements of process efficiencies and Dp.
- **Object:** The evaluation focuses on the *Artifact Itself*—the workflow—and the *Construction Process*, which includes the methodologies used for simulation and analysis.
- **Ontology:** Based on *Realism*, the workflow is seen as having concrete, measurable impacts

on PdM development processes that can be objectively analyzed and improved.

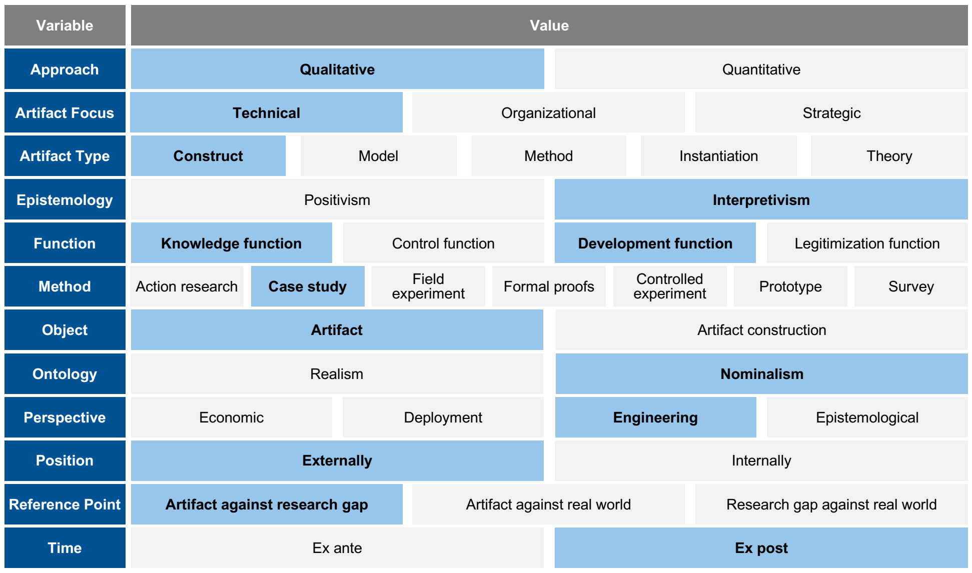
- **Perspective:** Evaluated from an *Engineering Perspective*, focusing on the technical efficiencies and process improvements provided by the workflow.
- **Position:** Evaluation is conducted *Internally* by the development team and *Externally* by independent reviewers, ensuring comprehensive validation of the workflow's effectiveness.
- **Reference Point:** Assessed against the *Research Gap* in efficient PdM development and its performance compared to traditional methods in *Real-World Applications*.
- **Time:** Includes both *Ex Ante* evaluations to predict its potential impacts and *Ex Post* evaluations to assess actual performance improvements post-implementation.

The Component Creation Workflow is a vital advancement in PdM development, introducing a structured, data-driven approach that significantly enhances the efficiency and precision of component creation. Its systematic evaluation demonstrates its superiority over traditional methods, marking it as a key innovation in the field of PdM.

3.3.5 Artifact Analysis of the Formal Descriptive Attributes System

The fifth artifact, **Formal Descriptive Attributes System**, is a sophisticated system designed to describe objects at all levels of development—model, module, and component—based on their input/output characteristics using a tree-graph formal attribute system. This system enhances the functionality of the previously introduced Component Creation Workflow (Artifact 4) by offering automated comparison and recommendation options for PdM model objects, accelerating the execution of the workflow. It is qualitatively evaluated through its application to real-world use cases. The evaluation is conducted ex post, and the reference point is against the identified research gap. Figure 17 illustrates the classification of this artifact according to the morphological field.

- **Approach:** This artifact employs a *Qualitative* approach, focusing on the subjective assessment of object attributes and their alignment within the system to ensure accurate comparisons and recommendations.
- **Artifact Focus:** The focus is *Technical*, as it deals directly with the properties and specifications of PdM model objects.
- **Artifact Type:** Classified as a *Construct*, this system provides a formal structure for defining and organizing the attributes of various components in a hierarchical manner.
- **Epistemology:** Reflecting an *Interpretivist* stance, the system is developed based on the



Variable	Value						
Approach	[✓ Qualitative]	Quantitative					
Artifact Focus	[✓ Technical]	Organizational	Strategic				
Artifact Type	[✓ Construct]	Model	Method	Instantiation	Theory		
Epistemology	Positivism	[✓ Interpretivism]					
Function	[✓ Knowledge function]	Control function	[✓ Development function]	Legitimization function			
Method	Action research	[✓ Case study]	Field experiment	Formal proofs	Controlled experiment	Prototype	Survey
Object	[✓ Artifact]	Artifact construction					
Ontology	Realism	[✓ Nominalism]					
Perspective	Economic	Deployment	[✓ Engineering]	Epistemological			
Position	[✓ Externally]	Internally					
Reference Point	[✓ Artifact against research gap]	Artifact against real world	Research gap against real world				
Time	Ex ante	[✓ Ex post]					

Figure 17: Classification of the fifth artifact, *Formal Descriptive Attributes System*, according to the morphological field for DSR artifact classification (own figure).

nuanced understanding of how attributes influence object functionality and integration within PdM models.

- **Function:** It serves the *Knowledge Function* by enhancing understanding of object characteristics and the *Development Function* by aiding the integration and application of these objects in real-world settings.
- **Methods:** The evaluation method includes *Case Studies*, applying the system to actual use cases to assess its functionality and impact.
- **Object:** The evaluation focuses on the *Artifact Itself*, examining how well the descriptive system functions in describing and comparing PdM objects.
- **Ontology:** Grounded in *Nominalism*, the system postulates that the properties and classifications it describes are constructed based on consensus and practical utility rather than existing as inherent qualities.
- **Perspective:** Evaluated from a *Technical Perspective*, considering the system's ability to accurately and effectively describe and link PdM model objects.
- **Position:** The evaluation is conducted *Externally*, by users and experts not involved in the system's development, to ensure objective assessment.
- **Reference Point:** Assessed against the *Research Gap*, focusing on the system's capability to improve PdM model development and integration.
- **Time:** The evaluation is *Ex Post*, conducted after the implementation of the system to determine its effectiveness in operational environments.

The Formal Descriptive Attributes System represents a critical enhancement in the domain of PdM, providing a robust and structured approach to describing and integrating PdM components. Its innovative use of a tree-graph structure for attribute management supports enhanced decision-making and model optimization, making it an invaluable tool in the field of PdM.

3.4 Ethical Considerations: A Value Sensitive Design Approach

This work adopts the Value Sensitive Design (VSD) framework to ensure that ethical considerations are integrally incorporated throughout the development of PdM models. VSD is a methodological approach that addresses the human values in technology design systematically and principled, involving the assessment of user values, ethical implications, and societal impact (Friedman and Hendry, 2019).

Conceptual Investigations involve the identification and clarification of stakeholders' values and the ethical issues inherent in PdM technologies. In this phase, the work considers the values of fairness, accountability, and transparency as fundamentally important. This involves exploring theoretical frameworks to understand how these values might be impacted or supported by the technology developed.

In the conceptual phase of the VSD framework, this work identifies and examines the fundamental ethical values and dilemmas associated with the development of PdM models in industrial applications, particularly in the automotive sector. Key values considered include *privacy*, due to the sensitive data involved; *fairness*, addressing potential biases in data and model outputs; and *transparency*, ensuring that the workings and decisions of PdM models are understandable to users. These values are critically analyzed to guide the design and operational strategies of the *Predictive Maintenance Framework*. The evaluation considers how technological choices can affect these values, both positively and negatively, and seeks to balance them against the technical and business requirements of PdM systems. This inquiry sets the stage for integrating ethical considerations into the empirical and technical stages of the VSD process, ensuring that the PdM models developed are not only efficient and effective but also align with the broader societal and ethical standards.

Empirical Investigations relate to the real-world context in which the PdM models operate. This involves direct engagement with stakeholders through interviews, surveys, and observations to gather insights about their concerns, expectations, and values. The objective is to derive a nuanced understanding of the stakeholders' perspectives, which informs the design and development processes to better align with their needs and ethical standards (Pennock and Bodner, 2020).

Empirical investigations in this work involve direct interactions with key stakeholders, including engineers, maintenance personnel, and management teams within the automotive industry, as well as end-users potentially affected by PdM systems. The aim is to gather qualitative and quantitative data through methods such as surveys, interviews, and participatory workshops. These activities are designed to collect insights into stakeholders' perceptions, expectations, and potential concerns regarding the ethical dimensions of PdM model implementation, such as data privacy, model fairness, and transparency. This data collection is crucial for understanding how stakeholders perceive the benefits and risks associated with PdM models, as well as their expectations for ethical accountability. The findings from these investigations inform the technical development of PdM models, ensuring that they are designed with a user-centered approach that respects and incorporates stakeholder values and ethical considerations into the system architecture and functionalities.

Technical Investigations focus on integrating and operationalizing the values identified in the conceptual and empirical phases into the PdM models. This includes developing features that enhance transparency, such as explainability interfaces, and mechanisms to ensure the accuracy and fairness of the predictive outputs (Doshi-Velez and B. Kim, 2017). It also involves iterative testing to verify that the model adheres to the ethical standards set forth in the earlier phases.

Technical investigations in this work focus on the application of ethical considerations identified in the earlier conceptual and empirical phases to the design and development of PdM models. This involves the implementation of technologies and strategies that enhance data privacy, such as encryption and access control mechanisms, to safeguard sensitive information processed by PdM systems. Additionally, algorithms are designed to be fair, avoiding biases that could lead to unfair predictive outcomes. Techniques such as algorithmic auditing and transparency features are also incorporated to ensure that the PdM models are not only effective but are understandable and explainable to users. These technical solutions are iteratively refined based on continuous stakeholder feedback, ensuring that the PdM models align with ethical standards and stakeholder values. Moreover, the development process adheres to best practices in software engineering, emphasizing modularity and reusability, which supports the agile and distributed development environments described in this work (Sommerville, 2011).

Integration and Iteration - Finally, VSD is an iterative process that requires ongoing evaluation and refinement. This work includes continuous feedback loops between the empirical findings and technical solutions to adapt the PdM models as new ethical challenges arise and as stakeholder values evolve. This iterative process ensures that the PdM models remain relevant and ethically sound as they scale and as industry standards change.

The integration and iterative refinement of ethical considerations are central to the ongoing development process of PdM models within this work's framework. By continuously cycling back to the insights gained from conceptual, empirical, and technical investigations, the development process adaptation incorporates stakeholder feedback and evolving ethical standards. This iterative loop ensures that the PdM models not only comply with initial ethical assessments but also respond dynamically to new challenges and stakeholder needs as they arise. Regular updates to the models and their underlying processes, guided by a structured feedback mechanism, enable the adaptation of PdM models to reflect ethical advancements and stakeholder expectations. This proactive approach to integration supports the sustainability of the PdM models, ensuring their long-term viability and ethical alignment in a rapidly changing technological landscape.

The application of VSD in this work ensures a stringent ethical framework, fostering the development of PdM models that are not only technically robust but also ethically responsible, aligning with the core values of all stakeholders involved.

3.5 Conclusion of the Methodological Overview

The methodology of DSR is adopted as the primary research approach for this work, due to its robust alignment with real-world environments and its provision of solid validation methods that effectively leverage these settings. The automotive industry, characterized by its diverse applications and the extensive availability and variability of data, presents an ideal environment for the development and validation of the artifacts described in this thesis.

The research environment within the automotive industry not only supports but also enhances the development of the artifacts by providing a rich, data-driven context. This setting allows for the practical application and thorough testing of the theoretical models and frameworks developed, ensuring that the artifacts are not only theoretically sound but also practically viable.

Each of the five artifacts developed in this work is thoroughly classified according to the morphological field established by Cleven, Gubler, and Hüner (2009). This classification ensures that each artifact is developed and assessed through a structured and systematic approach, adhering to the principles of DSR. Such a methodical classification aids in maintaining a consistent application of research processes across all artifacts and provides a clear interpretation framework for the insights gained during their development and evaluation.

Ethical considerations in this work are addressed through a VSD approach, which is integral to ensuring the ethical rigidity of the research methods and outcomes. This approach emphasizes the consideration of human values in a comprehensive manner throughout the entire design and implementation process. By incorporating ethical deliberations from the onset, this work aims to produce artifacts that are not only effective but also align with broader societal and ethical standards.

In summary, the methodological framework of DSR, combined with a well-suited research environment and a thorough classification system, enables the creation of innovative and practical artifacts within the automotive industry. The integration of VSD ensures that these developments are conducted with a strong ethical foundation, enhancing the reliability and applicability of the research outcomes. The stringent methodological approach and ethical rigidity embedded in this work exemplify a comprehensive and responsible research practice in the field of PdM.

Chapter 4

Execution

This chapter represents the culmination of the theoretical foundations laid in the literature review and the methodologies discussed previously, transitioning into the practical execution of the PdM Component Repository. The subsection 4.1 *Use Case Description Methodology*, illustrates the approach for detailing use cases critical to the development of PdM models. The significant findings of this research are comprehensively documented and published by Wolf, Sielaff, and Lucke (2023). This publication summarizes the key outcomes and advancements achieved during the course of the investigative efforts.

Subsequent sections 4.2 discuss the creation of a *Component Repository and Interface Standardization*, emphasizing modular design and interoperability. The chapter then explores the *A Priori Component Analysis for Reusability* in subsection 4.3, a novel approach to assess component reusability in early stages. This is followed by a detailed description of the *Workflow for Component Creation and Comparison* (subsection 4.4), and the *Descriptive Attributes System for Components* (subsection 4.5), which collectively enhance the efficiency and effectiveness of PdM model development. The chapter concludes with a summary of these key contributions, reflecting on how they collectively advance the field of PdM in the *Conclusion* subsection 4.6.

4.1 Use Case Description Methodology

In the quest to enhance the efficiency of PdM model development, especially within the framework proposed in this work, it is essential to have a comprehensive understanding of the entire development process. This understanding not only helps a smoother workflow but also ensures that the foundational elements of PdM model creation—such as stakeholder engagement, data acquisition, and requirement specification—are adequately addressed. To this end, this subsection introduces a standardized use case description approach, designed to formalize the initial phase of PdM development. By establishing a structured method for gathering critical inputs from stakeholders, collating necessary data, and mapping out precise requirements, this approach lays the groundwork for a more structured and effective development process. It represents a vital first step in the realization of robust PdM models, aligning with the overarching goals of the proposed framework to support and optimize the PdM development journey.

The methodologies and outcomes presented in this work are grounded in the practical, real-world context of an automotive OEM. The developed formal and standardized description model for PdM use cases, as detailed herein, is thoroughly validated through application to actual PdM scenarios for vehicle components. This validation process involved the analysis and examination of diverse PdM use cases within the automotive sector, ensuring that the proposed description model not only aligns with theoretical expectations but also meets the practical requirements and complexities encountered in the industry. Such an approach underscores the relevance and applicability of the research findings, demonstrating their potential to significantly enhance the development and implementation of PdM solutions across the automotive industry (Wolf, Sielaff, and Lucke, 2023).

The increasing recognition of PdM solutions by OEMs marks a paradigm shift towards leveraging technological advancements for competitive advantage, primarily through enhanced product reliability and reduced failure risks (Prytz, 2014; Carvalho et al., 2019). This is particularly evident in the automotive sector, where traditional maintenance strategies—such as scheduled checkups and oil changes—are gradually being supplemented by data-driven approaches. Modern vehicles, equipped with an array of sensors for various functions,[1] provide a rich data source processed by ECUs. These advancements, coupled with the extensive data infrastructure of automotive OEMs and the potential for scaling effects due to large vehicle fleets, set a conducive groundwork for PdM.

Despite the availability of data and the significant advancements in AI and ML technologies enhancing the applicability of PdM solutions, the development process remains complicated, time-consuming, and highly interdisciplinary. Common challenges include misunderstandings of use cases and communication barriers among development partners, often leading to redundant efforts in developing solutions for similar components. Addressing these challenges calls for novel approaches to update and enhance the PdM development process. A vital strategy, as

[1]e.g., Electronic Stability Program (ESP) and Heating, Ventilation and Air Conditioning (HVAC) systems.

elaborated in this work, involves the introduction of a formal and standardized description model for PdM use cases. This model, designed to accompany the PdM solution throughout its development lifecycle, aims to advance cross-domain communication and the reuse of existing solutions, thereby significantly improving the efficiency of PdM solution development (Wolf, Sielaff, and Lucke, 2023).

In the process of developing a comprehensive description model for PdM use cases, an extensive literature review is undertaken to explore existing research and methodologies within the domain, particularly those relating to the automotive industry. Utilizing databases such as SCOPUS, Web of Science, and Google Scholar with keywords including `Predictive Maintenance`, `Automotive`, `Fleets`, and `Review`, two vital works are identified that provide contemporary insights into the scientific progress in PdM applications. Zonta (2020) presents a structured analysis of industrial PdM applications including J. Lee, F. Wu, et al. (2014) and Gunes (2014), noting a shift from engineering-focused research to a more methodological approach, yet highlighting the absence of a standardized cross-domain template for PdM use cases. Theissler et al. (2021) further this discourse by categorizing automotive PdM research from a ML perspective, yet also underscore the lack of focus on organizational and management aspects in the field. The review extends to frameworks and standardization efforts in PdM, revealing various approaches to use case descriptions across industrial AI applications (Cockburn, 2012; De Souza and Cavalcanti, 2016), but none adequately addressing the specificities of automotive PdM in a standardized manner. Notably, works like those of Bertolino et al. (2008) and Grambau, Hitzges, and Otto (2019) attempt to formalize use cases and propose frameworks for integrating multiple data sources in PdM models, yet do not offer a unified framework tailored for automotive PdM applications. This literature review underscores a significant gap in the standardized description of PdM use cases within the automotive sector, a gap this work aims to bridge.

The cornerstone of this work's framework is the establishment of a standardized use case description, designed to accompany the PdM solution throughout its entire development lifecycle. Recognizing the critical importance of fostering a common understanding among stakeholders and promoting the reuse and adaptation of existing PdM solutions, the framework initiates with a high-level use case description. This initial phase is geared towards cultivating a mutual comprehension of the use case among all stakeholders, which is vital for directing the subsequent detailed development of the model, particularly regarding input and output data specifications. Figure 18 displays the CRISP-DM, which is used as a structure. As the development progresses,

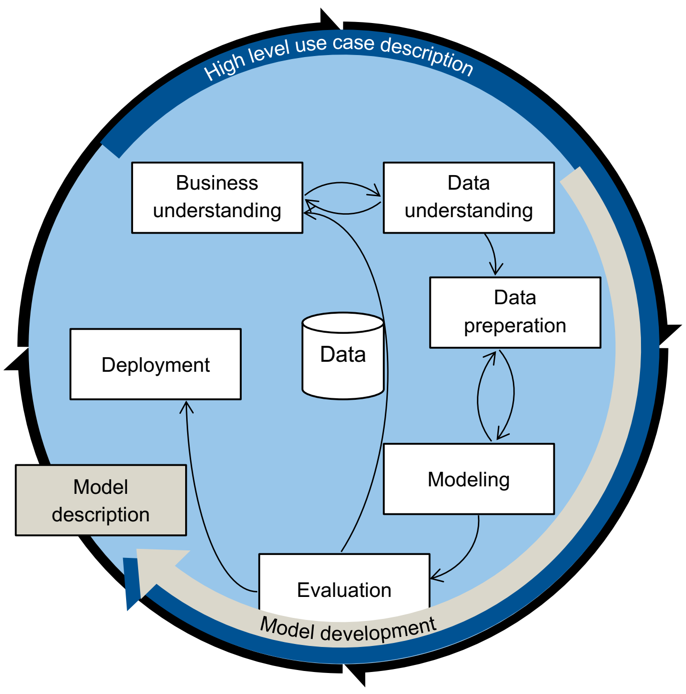


Figure 18: Phases of use case description and model development based on the Cross Industry Standard Process for Data Mining (CRISP-DM) process model, including business and data understanding, data preparation, modeling, evaluation, and deployment, according to Wirth and Hipp (2000).

the high-level description undergoes continuous refinement to ensure it evolves in tandem with the detailed specifications, ultimately forming a comprehensive characterization of the PdM solution.

The high-level use case description template, drawing inspiration from the business model Canvas (Zolnowski and Böhmann, 2014), is thoroughly crafted to facilitate rapid and interdisciplinary comprehension and application. This template, displayed in figure 19 presents a series of fields, each thoroughly designed to capture essential aspects of the use case, from organizational details to technical system descriptions and customer objectives. The initial field (field no. 0) is dedicated to organizational specifics, such as contact information, date, and versioning, crucial for document management and historical tracking. Subsequently, field no. 1 looks into the technical system associated with the product or component in focus, employing a descriptive chain methodology to ensure precision and comprehensibility in outlining technical challenges and system intricacies.

The objective of the use case, as perceived by the customer (field no. 2), aims to articulate the envisioned solution's impact, whether as an enhancement to existing services or as an innovative offering, thereby ensuring that the development trajectory remains aligned with customer value. This customer-oriented perspective is further encapsulated in the use case title (field no. 3), which concisely conveys the component and objective, accelerating straightforward classification and communication across departments.

The comprehensive nature of the use case is brought to the forefront in the short description (field no. 4), which integrates unaddressed information from previous fields, including insights into the wear mechanism or technical specifications, supplemented by visual aids where applicable. This section ensures a nuanced understanding of the use case, enhancing clarity and context. The template also addresses the origins and characteristics of required input data (field no. 5) without delving into technical specifics prematurely, setting the stage for a detailed model development phase. The anticipated model's capabilities, deployment considerations, and algorithmic requirements are encapsulated in field no. 6, laying the groundwork for informed development strategies.

Furthermore, the output data expected from the model is specified in field no. 7, detailing the information to be generated for effective use in subsequent processes. The penultimate field (no. 8) aggregates necessary competencies, identifying both the skills required and potential contributors to the use case's realization. Lastly, an optional field (at the bottom right) allows for the inclusion of additional remarks or references, ensuring that no critical insight or cross-connection is overlooked in the documentation process.

This structured approach, articulated through the template, ensures a comprehensive, clear, and systematic documentation of PdM use cases, aligning technical specifications with customer expectations and allowing effective communication and collaboration among stakeholders.

Transitioning to model development demands a more granular focus on the data specifics, warranting a detailed documentation format. This phase involves outlining the required input and output data in a tabular form, ensuring flexibility to accommodate varying data sources. Detailed descriptions include data source, format, security classification, and storage specifics, among others. This in depth documentation aids a clear understanding of the data landscape, central for developing robust PdM models tailored to specific use cases.

The validation of the proposed standardized use case description model is thoroughly carried out through the application to an existing PdM scenario, specifically aimed at enhancing the understanding of vehicle brake noises. This real-world use case served as a foundation for both direct application of the model and for gathering comprehensive feedback from domain experts, including use case owners, which is complemented by a self-evaluation of the use case description process. The use case involved categorizing brake noises into clusters, manually labeling acoustic

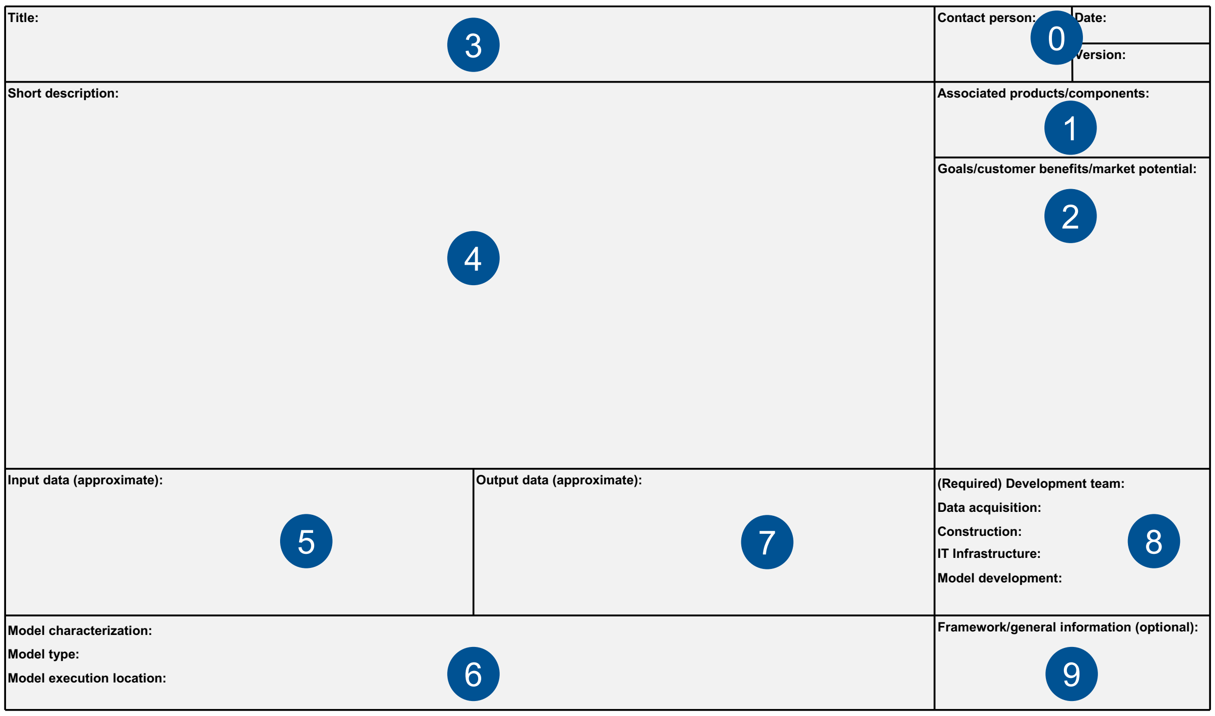

Title:
3
Contact person:
0
Date:
Version:
Short description:
4
Associated products/components:
1
Goals/customer benefits/market potential:
2
Input data (approximate):
5
Output data (approximate):
7
(Required) Development team:
Data acquisition:
Construction:
IT Infrastructure:
Model development:
8
Model characterization:
Model type:
Model execution location:
6
Framework/general information (optional):
9

Figure 19: *Use Case Description* template with numbered fields for organizational details, technical system descriptions and customer objectives (own figure).

data for training an ANN, and employing this network to classify unlabeled fleet data.

The template's efficacy is demonstrated through its application to this use case, detailing components, objectives, customer benefits, and the data-driven methodology employed for noise classification. Feedback from various stakeholders highlighted the template's utility in facilitating a clear understanding of technological contexts, data requirements, model specifics, and project resources. Domain experts appreciate the structured presentation of critical information, which aided in defining responsibilities, assessing project feasibility, and guiding team formation and management. This validation process underscores the template's comprehensiveness and its ability to foster effective communication and understanding among cross-functional teams, ultimately affirming the proposed approach's validity and effectiveness in the context of PdM solutions development.

This work successfully introduces a standardized model for the description of PdM use cases, designed to reorganize the collection, presentation, and documentation of essential information across various development phases. This model not only eases the communication and collaboration among diverse stakeholders involved in PdM solutions but also ensures the continuity and relevance of the use case documentation throughout the development lifecycle, from conceptualization to deployment. By aligning with established data mining process models like CRISP-DM, the description model demonstrates a significant potential to advance the adaptation of existing PdM solutions to new components, thereby enhancing the efficiency and effectiveness of PdM initiatives. The validation within an automotive sector use case further attests to the model's applicability and value, prompting considerations for broader applications across different industry sectors. Future research will pursue additional validations within various domains, aiming to refine and integrate the description model into a comprehensive process model. This model will address the nuances of multidisciplinary collaboration, process and stakeholder distribution, and the evolving landscape of technological development processes, particularly in cloud-based environments.

4.2 Component Repository and Interface Standardization

In the development of this work's PdM component repository, inspiration is drawn from various principles and components inherent in existing frameworks from the domains of ML, PdM, MLOps and established Software Engineering.

The primary focus of MLOps (Zaharia et al., 2018; Biewald, 2023; Tagliabue et al., 2023) frameworks is on ensuring efficient deployment, maintenance, and continuous improvement in production environments. By automating and standardizing the manual and time-intensive aspects of ML projects, MLOps aims to significantly reduce the time required for deploying ML models (Hewage and Meedeniya, 2022). This remodeled process not only enhances the speed of MD but also plays an important role in accelerating the overall ROI for organizations (Testi et al., 2022).

While MLOps primarily focuses on the reorganized deployment and management of ML models in production, Automated Machine Learning (AutoML) (Ali, 2024; Jin, Song, and Xia Hu, 2018; Feurer et al., 2020; AutoML, 2024) takes a different approach, aiming to minimize the time spent on the actual development of the model, or to enable model development for developers with almost no ML knowledge. Unlike MLOps , which often assumes the model as a given entity, AutoML automates crucial aspects of the development process. This includes tasks such as Feature Encoding (FE), Model Building (MB), and hyperparameter optimization. By doing so, AutoML empowers individuals with varying levels of ML expertise to create effective models, thereby significantly reducing the time required from the conceptualization of a model to its deployment (X. He, K. Zhao, and Chu, 2021).

It's essential to recognize that there are situations where deployment and model selection may not be fully automated. In such cases, the utilization of ML frameworks like *PyTorch* (Paszke et al., 2019), *TensorFlow* (Abadi et al., 2016), *Scikit-learn* (Pedregosa et al., 2012) becomes essential. These frameworks specialize in expediting the implementation of ML models, offering developers a versatile set of predefined functions and optimizations. By leveraging these tools, even in scenarios where certain tasks require manual intervention, substantial time savings can still be achieved, contributing to the efficiency of the overall ML workflow.

For steps that cannot be accelerated through ML frameworks, MLOps, or AutoML frameworks, integrating proven methods from established software development, such as Data Versioning, Directed Acyclic Graphs (DAGs) and modularization, offers a time-saving alternative. These established practices contribute to making development processes more efficient, creating a synergistic connection between modern ML technologies and traditional software engineering methods.

Shaping the PdM component repository, inspiration is not only from established MLOps, AutoML, and established software engineering principles but also from two distinctive frameworks: *PyWatts* (Heidrich et al., 2021) and *Kedro* (Alam, 2024). *PyWatts*, prioritizing end-to-end workflows as pipelines, amplifies reusability, while *Kedro* serves as a Python framework for crafting reproducible, maintainable, and modular Data Science (DS) code. In contrast to straightforward categorization into MLOps or AutoML , both *PyWatts* and *Kedro* defy such constraints by serving as overarching frameworks that integrate these methodologies, a characteristic mirrored in the *Predictive Maintenance Framework*.

This works developed framework, strategically integrates principles and components from all the presented frameworks. Its overarching objective is to not only accelerate MD or discovery but to minimize the entire development process of new use cases. This comprehensive approach proves especially valuable in scenarios where uncertainties persist, be it in terms of data compatibility, project goals, or unfamiliar use cases. By reducing the time required for launching, development, and exploration, the component repository framework empowers teams to navigate uncharted territories in PdM with agility, fostering a culture of fast innovation.

Another distinction between this work's framework to previously introduced ones lies in its tailored focus on the development of PdM within the automotive domain. This specialization involves the incorporation of elements aligning with automotive requirements, such as adaptations to diverse vehicle systems, data sources or PdM algorithms. By concentrating on the automotive sector and PdM, the likelihood of reusing entire pipelines or sub-pipelines is heightened, resulting in substantial time savings.

Figure 20 presents a sophisticated application of CBSE modularization principles within the context of a data-driven value creation model, specifically tailored for a PdM model. This modular system is the culmination of an extensive literature review, as detailed in Subsection 2.6, coupled with empirical observations gathered from the artifact environment in this study. The ensuing discussion offers an in-depth exploration of this modular system, encompassing several key aspects: the general process workflow, the diverse types of value creation models and their associated process exit points, the requisite roles of expertise within individual modules, and the overarching reach of the module system, culminating in its integral contribution to the component repository.

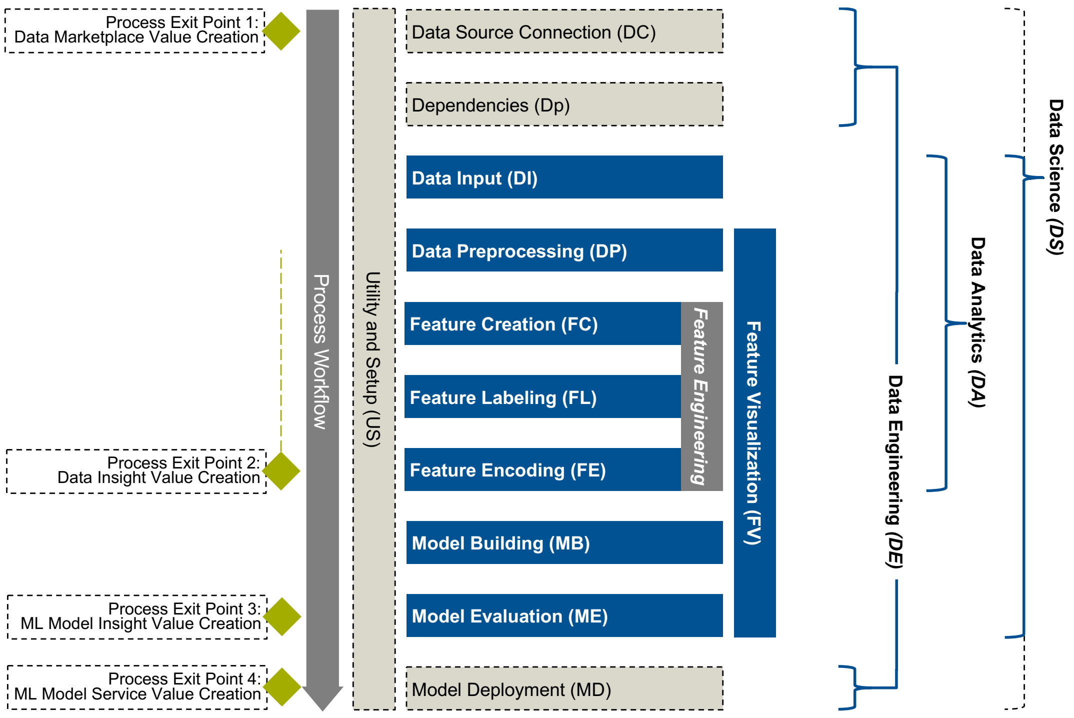


Figure 20: Modular system of the *Component Repository* with four types of value creation models and process exit points (green), process modules (DS modules in blue, Data Engineering (DE) modules in grey), and the roles of expertise for the Modules: DE, Data Analytics (DA) and DS (own figure).

The general process workflow, as depicted in the figure, is a testament to the systematic and methodical approach inherent in CBSE. It maps out a clear trajectory from the initial data sourcing to the final MD, ensuring a coherent and efficient progression through various stages of model development. This workflow is not only instrumental in streamlining the development process but also in enhancing the reproducibility and scalability of PdM applications.

The figure further illustrates various types of value creation models, each marked by distinct process exit points. These exit points signify the transition from one phase of value creation to another, reflecting the multifaceted nature of value generation in PdM. The identification and analysis of these exit points are crucial for understanding how different stages in the workflow contribute to the overall value creation process.

Another critical aspect highlighted in the figure is the roles of expertise required by individual modules. These roles encompass a range of skills and knowledge areas, underscoring the interdisciplinary nature of PdM model development. The expertise required at each module ensures that the development process is not only technically sound but also aligned with the specific needs and objectives of the PdM application.

Lastly, the figure encapsulates the comprehensive scope of the module system, illustrating its extensive reach and applicability in the realm of PdM. This system's breadth and depth are indicative of its potential to significantly enhance the efficiency and effectiveness of PdM model development. Moreover, its relevance to the component repository is particularly noteworthy, as it provides a structured framework for organizing and accessing various components essential for PdM applications.

In summary, Figure 20 not only serves as a visual representation of the modular system but also as a conceptual map guiding the development of PdM applications. Its detailed breakdown into workflow, value creation models, roles of expertise, and overall scope offers valuable insights into the complexities of CBSE in the context of PdM.

The initiation of the process workflow is marked by the **Data Source Connection (DC)** phase, a critical juncture that encompasses all necessary operations to establish both technical access and requisite permissions. This phase is multifaceted, involving the integration of various data sources, which may include cloud storage areas, in-house databases, third-party data, or customer-specific datasets. The DC process is not limited to mere data retrieval; it also involves handling static data files, potentially decrypting data, and managing access licenses or software requirements.

A central aspect of the DC phase is its role in interfacing with subsequent process steps. It lays the groundwork for the entire workflow, ensuring that data flows seamlessly into the following stages. Moreover, the DC phase has a significant impact on the overall efficiency of data platform operations. This includes considerations related to maintenance efforts and platform costs, which are vital for the sustainable management of the data infrastructure.

Furthermore, the DC phase bears the responsibility for the correctness and integrity of the supplied data. This is fundamentally important, as the quality of data directly influences the effectiveness of downstream processes, such as Data Preprocessing (DP), feature engineering, and MB. Ensuring data accuracy and reliability at this early stage is essential for the success of the entire PdM model development process.

The **Dp** module plays a crucial role in the PdM model development process, acting as the cornerstone for establishing a conducive working environment. This module is not merely a preparatory step; it is an integral component that ensures the availability and readiness of all necessary resources and tools required for model construction.

At its core, the Dp module is responsible for setting up a robust 'working bench', a metaphorical term that encompasses the entire spectrum of the development environment. This includes the provision of essential coding libraries, which are the building blocks for any data-driven model. These libraries may range from general-purpose programming libraries to specialized *Predictive Maintenance Frameworks*, each serving a specific function in the model development process.

In addition to technical resources, the Dp module also emphasizes the importance of expert know-how. This aspect underscores the interdisciplinary nature of PdM model development, where expertise in various domains such as data science, engineering, and domain-specific knowledge converge. The integration of this expert knowledge is vital for navigating the complexities of MB, ensuring that the developed models are not only technically sound but also aligned with the specific requirements and nuances of the application domain.

Overall, the Dp module is a testament to the multifaceted approach required in PdM model development. It highlights the need for a well-equipped and knowledge-rich environment, where the synergy of technical tools and expert insights paves the way for the creation of effective and efficient PdM models.

The **Data Input (DI)** module serves as a critical bridge between the DC module and the subsequent stages of the PdM model development process. Its primary function is to render the data, sourced from the DC module, both accessible and amenable for further processing. This module plays a vital role in ensuring that the data is not only retrievable but also in a state that is conducive for analysis and MB.

A significant challenge addressed by the DI module is the optimization of data size. This involves a thorough process of data selection and reduction, aimed at retaining only the most relevant and essential data for the development and training of the ML model. The rationale behind this is twofold: firstly, to enhance the efficiency of the model development process by minimizing computational load and storage requirements, and secondly, to improve the performance of the ML model by focusing on high-quality, pertinent data.

The process of data reduction in the DI module is not a mere truncation of the dataset but a strategic and informed decision-making process. It involves identifying and filtering out redundant, irrelevant, or low-quality data, thereby ensuring that the dataset fed into the model is optimized for both accuracy and efficiency. This step is important, as the quality and relevance of the input data directly influence the usefulness of the ML model, impacting its ability to generate accurate and reliable predictions in a PdM context.

The **DP** module is a fundamental component in the PdM model development workflow, tasked with refining and transforming raw data into a format that is suitable for in-depth analysis and model training. This module encompasses a wide array of preprocessing steps, each tailored to enhance the quality and utility of the data.

At the technical level, DP involves standardizing data formats and decoding data to ensure uniformity and accessibility. This step is crucial for aligning disparate data sources and formats, which is a common challenge in PdM applications where data may come from varied sensors, systems, or databases. The standardization process aids smoother integration and analysis of data, thereby improving the efficiency of the model development process.

Beyond technical adjustments, DP also includes combinatory steps that are essential for creating a comprehensive dataset. This involves merging multiple datasets with different structures or formats, a process that requires careful alignment and synchronization of data. Such integration is vital for developing a holistic view of the maintenance needs and patterns, enabling more accurate and effective PdM models.

Another critical aspect of DP is the selective removal of unnecessary or insignificant data. This data reduction is not arbitrary but a strategic decision aimed at focusing on the most relevant and impactful data. However, it is essential to thoroughly document any data removal in the metadata. This documentation is essential for maintaining transparency and traceability in the data processing workflow, preventing potential misinterpretations or false conclusions that might arise from the absence of certain data points.

To sum up, the DP module plays a vital role in ensuring that the data is not only technically sound and integrated but also strategically refined and documented. This comprehensive approach to DP lays a solid foundation for the subsequent stages of PdM model development, ultimately contributing to the creation of more accurate, reliable, and efficient PdM solutions.

The **Feature Visualization (FV)** module is an integral part of the PdM model development workflow, designed to generate value by enabling the visualization of both raw and preprocessed data, as well as intermediate and final results. This module serves as a main interface between the developer and the data model, allowing a deeper understanding and interaction with the model at various stages of its development. The necessity of FV spans the entire spectrum of the model development process, from the initial access to raw data provided by the DI module to the final stages of Model Evaluation (ME).

FV is essential not only as a standalone module but also in its parallel operation with other functional modules. From the moment raw data becomes accessible, FV provides invaluable insights into the data's characteristics and the model's behavior. This continuous interaction is fundamental for iterative model refinement and for ensuring that the model development is aligned with the specific requirements and objectives of the PdM application.

The role of FV in bridging the gap between data and model developers cannot be overstated. By offering visual representations of data and model outputs, FV enhances the interpretability and transparency of the model, making it more accessible and comprehensible to those involved in its development and application. This accessibility is particularly important in complex PdM scenarios, where understanding the nuances of data and model behavior is key to successful implementation.

In essence, the FV module is a vital component that enriches the PdM model development workflow. It not only aids in visualizing and understanding data but also plays a significant role in enabling effective communication between the data and the model, thereby enhancing the overall quality and effectiveness of PdM solutions.

The **Feature Creation (FC)** module, a vital component of FE in PdM model development, encompasses the creation and enhancement of features from data. This module is not limited to merely modifying the structure of preprocessed data; it extends to the combination of various data sources, potentially including unformalized or static external data such as expert knowledge or common knowledge. The essence of FC lies in its ability to transform raw data into meaningful features that significantly contribute to the model's predictive power.

The process of feature generation in FC involves sophisticated methods that go beyond basic data manipulation. It includes the application of automated FC tools, which utilize a comprehensive list of feature generation methods to produce model-relevant features. These tools are adept at sifting through extensive datasets to identify and extract features that are most pertinent to the predictive model's objectives (Selçuk et al., 2021).

Moreover, FC plays an important role in enhancing the model's performance by integrating advanced ML algorithms. These algorithms are instrumental in selecting high-performant features, thereby ensuring that the model is not only accurate but also efficient in its predictions. The integration of such algorithms in FC is a testament to the module's importance in the overall feature engineering process, particularly in the context of PdM (Perner, 2019).

In summary, the FC module stands as a cornerstone in the development of PdM models. Its ability to create and refine features from a diverse range of data sources, coupled with the application of advanced feature generation and selection methods, underscores its significance in building robust and effective PdM solutions.

The **Feature Labeling (FL)** module, as a crucial part of Feature Engineering in PdM model development, focuses on the labeling of generated data. In supervised learning algorithms, labeling is essential as it signals the target value for both training and test data. The process of labeling, especially in classification and regression models, requires the label to be a one-dimensional numeric value. However, the creation of these labels is far from trivial, particularly in the context of pattern recognition models where the occurrence of a pattern must be accurately labeled, requiring a complex analysis of the available data.

In the FL module, the strategy for what to predict in a supervised ML model is defined based on the preliminary results of the preceding modules. This involves a thorough process of determining the most relevant and significant features to be labeled, which is essential for the model's accuracy and effectiveness. The complexity of FL is often amplified in the industrial context of PdM models, where the complications of real-world applications add layers of challenges to this task (Rosati et al., 2023).

Moreover, the FL module is characterized by its well-defined input and output interfaces. These interfaces are vital for maintaining the integrity and consistency of the labeling process, ensuring that the labels are correctly assigned and accurately reflect the target values intended for model training and evaluation. The importance of precise FL in PdM cannot be overstated, as it directly impacts the model's ability to make accurate predictions and, consequently, the effectiveness of maintenance strategies in industrial settings (Yuehua Liu et al., 2022).

Summing up, the FL module plays a central role in the feature engineering process of PdM models. Its responsibility for accurately labeling features, coupled with the challenges inherent in industrial applications, underscores its significance in the development of robust and reliable

PdM solutions.

The **FE** module, concluding the trio of FE modules in PdM model development, is dedicated to translating data from a human-readable and understandable format to a machine-readable format. This translation is essential for preparing the data for ML models, where machine-readability refers to the format required by these models for efficient processing and analysis.

Feature encoding involves various techniques, such as encoding categorical data types or normalizing features to a comparable range. This process is crucial as it ensures that the data is in a format that can be effectively utilized by ML algorithms. For instance, the encoding of categorical data into numerical values is a common practice in ML to enable the processing of non-numeric data types (Ferraro et al., 2020).

It is important to note that the FE module marks the first point in the model creation process where information is reduced. Unlike the previous modules, where information is either transformed or augmented by combining features and data sources, FE involves a certain level of information loss. Therefore, maintaining an overview of any information that might be lost during this process is crucial for the integrity of the model (De Santo et al., 2022).

The output of the FE module is typically a feature sample matrix with normalized and numeric values, ready to be processed by subsequent ML modules. However, it's also worth noting that some types of data visualization might require the encoding of features, further emphasizing the versatility and importance of this module in the PdM model development process.

In summary, the FE module plays a vital role in transforming data into a format that is conducive for ML models. Its ability to encode features effectively is key for the success of PdM applications, particularly in industrial contexts where the accuracy and reliability of predictions are fundamentally important.

The **MB** module in PdM model development encompasses the construction of predictive models, predominantly utilizing MLtechniques. This module is the crux of the model development process, where the actual predictive models are built, parameterized, and optimized for performance. A key aspect of MB is hyperparameter tuning, which involves adjusting the model's parameters to enhance its predictive accuracy and efficiency.

During the MB phase, various ML algorithms are employed and tested to determine the most effective approach for the specific PdM application. This selection process is based on the nature of the data, the desired outcome, and the computational resources available. The choice of algorithm can range from simple regression models to complex deep learning networks, each with its unique strengths and applicability (Bouabdallaoui et al., 2021).

Hyperparameter tuning in MB is a critical step that significantly impacts the model's performance. It involves experimenting with different combinations of parameters to find the optimal settings for the model. This process can be both time-consuming and computationally intensive but is essential for achieving the highest possible accuracy and efficiency in predictions (Lyubchyk et al., 2022).

To sum up, the MB module is a vital component in the PdM model development process. It involves the careful construction, parameterization, and optimization of ML models, ensuring that they are well-suited for the PdM tasks at hand. The success of the MB phase directly influences the effectiveness of the PdM solution in predicting and preventing equipment failures.

The **ME** module in PdM model development is distinct yet often intertwined with the MB module. While ME technically involves assessing the performance of the predictive models, its significance extends beyond mere technical evaluation. This module plays a crucial role in interpreting the prediction results within the technical environment and the business context of the model's application. Such an analysis is essential for understanding the broader implications of the model's deployment.

In ME, the focus is not only on the quantitative metrics of prediction quality, such as accuracy, precision, and recall, but also on the qualitative aspects of model performance. This includes evaluating how well the model's predictions align with real-world scenarios and the potential impact of these predictions on maintenance strategies and business operations (Vollert, Atzmueller, and Theissler, 2021).

Moreover, ME involves a thorough examination of the model's interpretability, especially in complex industrial contexts where decisions based on model predictions can have significant consequences. The ability to understand and explain the model's predictions is essential for gaining trust and acceptance among stakeholders (Che et al., 2021).

The output of the ME module often includes a detailed analysis of the model's performance, highlighting areas of strength and potential improvement. This analysis is critical for making informed decisions about the model's deployment and for identifying any adjustments or refinements that may be necessary to enhance its effectiveness in a PdM setting.

Summed up, the ME module is a vital component of the PdM model development process. It extends beyond technical evaluation to include a comprehensive analysis of the model's performance in the context of its intended application. This holistic approach to ME ensures that the deployed PdM models are not only technically sound but also practically relevant and aligned with business objectives.

The **MD** module is the final step in the PdM model development process, focusing on the deployment of the predictive models into a real-world environment. This module is critical as it transitions the model from a development setting to an operational one, where it can provide tangible benefits in PdM applications.

Model deployment involves integrating the predictive model into the existing technological infrastructure. This integration must be consistent to ensure that the model functions effectively within the industrial ecosystem. It often requires collaboration between data scientists, IT specialists, and domain experts to ensure that the model is correctly implemented and aligned with the technical and business requirements (A. Gorishti and K. Gorishti, 2022).

A key aspect of MD is ensuring that the deployed model can efficiently process real-time data and provide timely predictions. This is particularly important in PdM, where the ability to predict equipment failures in advance can significantly reduce downtime and maintenance costs. The model must be robust and scalable to handle varying data volumes and operational demands (Salierno et al., 2020).

Furthermore, MD also involves continuous monitoring and maintenance of the model to ensure its ongoing accuracy and relevance. This includes updating the model as new data becomes available, adjusting it to changing conditions, and troubleshooting any issues that arise during its operation.

In summary, the MD module is a fundamental phase in the PdM model development process, marking the transition of the predictive model from a theoretical construct to a practical tool. Successful deployment requires careful planning, collaboration, and ongoing management to ensure that the model delivers reliable and valuable predictions in a real-world setting.

The **Utility and Setup (US)** module in the PdM model development process encompasses the broad range of technical efforts that are not linked to a specific step in the model development lifecycle. This module spans across all other modules and is unique in that it does not have clear input and output interface standards. The US module includes aspects such as the development environment, team collaboration tools, and other supporting utilities that advance the model development process.

The US module is vital as it addresses the foundational aspects that enable efficient and effective model development. This includes setting up the development environment, which involves selecting and configuring the software tools, libraries, and platforms used for MB and ME. The choice of these tools is critical as it can significantly impact the efficiency of the development process and the quality of the final model (Lechuga et al., 2023).

Collaboration tools are another vital component of the US module. In a PdM project, team members often come from diverse backgrounds, including data science, IT, and domain-specific areas. Effective collaboration tools are essential for facilitating communication, sharing resources, and coordinating tasks among team members, thereby enhancing the overall productivity and success of the project (Shanin, Stupnikov, and Zakharov, 2019).

Moreover, the US module also encompasses the efforts to ensure that the model development process is scalable, reproducible, and adheres to best practices. This includes version control systems, documentation practices, and adherence to coding standards, which are essential for maintaining the quality and integrity of the model development process.

To sum up, the US module plays a foundational role in the PdM model development process. It provides the necessary infrastructure and tools that support the various stages of model development, from initial setup to final deployment. The effectiveness of the US module is key to the overall success and sustainability of PdM projects.

In the depiction of the modules in figure 20, those displayed in blue are predominantly code-centric, sharing similarities in appearance and handling, making them particularly amenable to this works repository approach, whereas the modules presented in gray are fundamentally rooted in architecture and infrastructure considerations.

In the context of PdM model development, process exit points are crucial stages where a data model successfully delivers tangible outcomes, marking significant milestones in data-driven value creation. These points not only signify the achievement of specific objectives within the model development lifecycle but also demonstrate the adaptability of the process to various project needs. They serve as key indicators for assessing progress, guiding resource allocation, and allowing the replication of successful practices across different projects, thereby enhancing the efficiency and impact of data-driven initiatives in PdM.

Having established the significance of process exit points in the PdM model development, the next step is a detailed exploration of the four distinct process exit points, each representing a critical phase of achievement and value creation within the data-driven development lifecycle.

The first process exit point, termed **Data Marketplace Value Creation**, hinges on the infrastructural access to the output of the initial module, DC. This exit point encapsulates the concept of platform access for third-party service providers to available data via a data marketplace. In the realm of PdM, this is particularly pertinent for the development of customer services by third-party suppliers. Various architectural models exist for this purpose, ranging from direct contact with data consumers to externalizing data distribution to a central data marketplace provider. Such architectures are instrumental in accelerating the efficient exchange and utilization of data in PdM, enhancing the development of customer-centric services and solutions (Rosati et al., 2023).

The second process exit point, **Data Insight Value Creation**, is vital in the PdM model development process. It focuses on the value generated through descriptive analysis, often in conjunction with FV. This exit point typically leverages the outputs of modules such as DP, FC, FL, and FE, particularly after the application of components from FV. Data insights, key for building predictive models, also hold intrinsic value by providing a deeper understanding of the data characteristics and potential implications. This process exit point underscores the general applicability of the model, highlighting that despite different objectives, the same process model can be effectively utilized, with the exit points varying based on the specific goals and outcomes desired in PdM (Pereira, 2020).

The third process exit point, **ML Model Insight Value Generation**, is centered around the outcomes of prescriptive statistical models, primarily achieved through the application of ML algorithms. This exit point is often the core of value generation in most PdM scenarios. Key functionalities such as RUL prediction, deviation detection, and maintenance need classification are all derived from these ML models. Additionally, the insights gained from these models can be utilized as features or labels in subsequent models, especially in pattern recognition use cases. This process exit point exemplifies the specific value derived from ML models in PdM, highlighting the tailored application of the general process model to different objectives, thereby underlining its broad applicability (Abdelli, Griesser, and Pachnicke, 2022).

The fourth process exit point, **ML Model Service Value Creation**, encapsulates the continuous operation of ML models, generating ongoing value through the use of live data. This exit point is critical in PdM models as it represents the service of having a perpetually running ML model, which continuously analyzes and interprets data to provide actionable insights. Such continuous operation is essential for real-time monitoring and decision-making, enabling proactive maintenance strategies and enhancing operational efficiency. The value generated at this exit point is not just in the predictive capabilities of the model but also in its ability to adapt and evolve with the incoming data stream, ensuring sustained relevance and accuracy in its predictions (Fausing Olesen and Shaker, 2020).

In the landscape of data-driven processes, particularly in PdM, the roles of DE, DA, and DS are distinct yet interconnected in the data-driven process, with DA focusing on extracting information from large datasets (Adeel Mannan et al., 2019), and DA concerned with the interpretation and analysis of data patterns (Othmane, Jaatun, and Weippl, 2018), each playing a crucial role in the overall effectiveness and efficiency of PdM processes.

Data Engineering primarily covers the modules of DC, Dp, and MD. This role demands proficiency in database management, data processing, and system integration, focusing on the infrastructure that supports data flow and storage.

Data Analytics concentrates on the modules from Data Input to the Process Exit Point 2, Data Insight Value Generation. This role requires skills in statistical analysis, data visualization, and the ability to derive meaningful insights from data, particularly in understanding and communicating the value generated from data up to the point of descriptive analysis.

Lastly, **Data Science** encompasses the entire process but with a focus on model creation from available data until ME. This role demands a deep understanding of ML, algorithm development, and advanced analytical techniques. The distinction between these roles lies in their specialized skill sets and focus areas - DE emphasizes the architecture that enables data availability, DA focuses on interpreting and visualizing data, and DS integrates these aspects to develop and evaluate predictive models, each contributing uniquely to the overarching goal of effective PdM.

The distinction of roles in DE, DA, and DS is crucial for designing a component repository tailored to individual modules, significantly enhancing reusability in PdM processes. By clearly defining these roles, the component repository can be structured to cater to the specific needs and skill sets of each domain. For DE, the repository can focus on components that permit efficient data ingestion, storage, and management. For DA, it can offer tools for data processing, visualization, and basic analysis, aligning with the role's focus on extracting insights from data. In DA, the repository can include advanced analytical models and ML algorithms. This role-based compartmentalization ensures that each component is optimized for its intended purpose, making it easier for professionals to find and reuse relevant components. It rationalizes the development process, as practitioners can quickly identify and deploy the tools that best fit their specific requirements, thereby reducing development time and enhancing the efficiency and effectiveness of PdM solutions.

4.3 A Priori Component Analysis for Reusability

The concept of *a priori* component reusability analysis in PdM applications plays an important role in enhancing development efficiency. This subsection explores methodologies for evaluating the potential reusability of components in PdM models, with a focus on the *a priori* aspect. This analysis differs from *a posteriori* analysis, which assesses reusability based on historical usage data. *A priori* analysis, in contrast, predicts the reusability potential of components without relying on their past usage, thereby streamlining the development process by identifying reusable components early. The literature review sheds light on existing metrics and methodologies for *a priori* analysis. The results section introduces novel metrics developed from these existing ones, tailored to the specific research environment in PdM. The conclusion synthesizes these findings and their implications for future PdM model development.

In the realm of PdM, reusability metrics are multifaceted, including both interface-based and code-inclusive evaluations of components. These metrics aim to provide insights into the potential reusability of a component, acting as necessary conditions to enhance the quality of a component repository sustainably. Historically, in Object-Oriented Software Engineering (OOSE), reusability metrics are categorized into two main types: White-Box components, which evaluate software components based on their source code (Koteska and Velinov, 2013), and Black-Box components, which assess components without their code (Boxall and Araban, 2004). The subsequent sections will discuss some of these metrics and evaluate their applicability in PdM.

Furthermore, the comprehensive analysis of component reusability significantly contributes to the evaluation of the component repository as a whole. This analysis is integral to addressing the research questions posed in this work. Specifically, the *a priori* analysis plays a vital role in designing the architecture of the component repository. By examining state-of-the-art reusability metrics, this subsection aims to synthesize custom component design principles. These principles are intended to reduce costs associated with enabling reusability and the actual reuse of components, thereby contributing to the overall efficiency and effectiveness of the PdM model development process.

This subsection offers a comprehensive review of the vital literature on reusability metrics within the PdM context. It methodically explores White-Box metrics, which examine components based on their intrinsic characteristics, encompassing both component and solution perspectives. Subsequently, it examines Black-Box metrics, evaluating components based on external attributes such as understandability, adaptability, and portability. Each metric category provides distinct insights into component reusability, essential for the effective management and enhancement of a PdM component repository.

In the component-centric approach of White-Box metrics, the primary focus is on the component itself. The objective is to evaluate the appropriateness of specific components for inclusion in a component repository or to pinpoint necessary modifications. Regular monitoring of a component's long-term value is instrumental in determining its compatibility with component-based methodologies. Metrics that track the frequency of component reuse and modifications shed light on the potential need for expanding a component with new functionalities (Koteska and Velinov, 2013).

Koteska and Velinov (2013) propose metrics to assess the reusability of White-Box components. One such metric quantifies reusability based on the proportion of reused code:

$$R = \frac{L_r}{L_r + L_n} \tag{4.3.1}$$

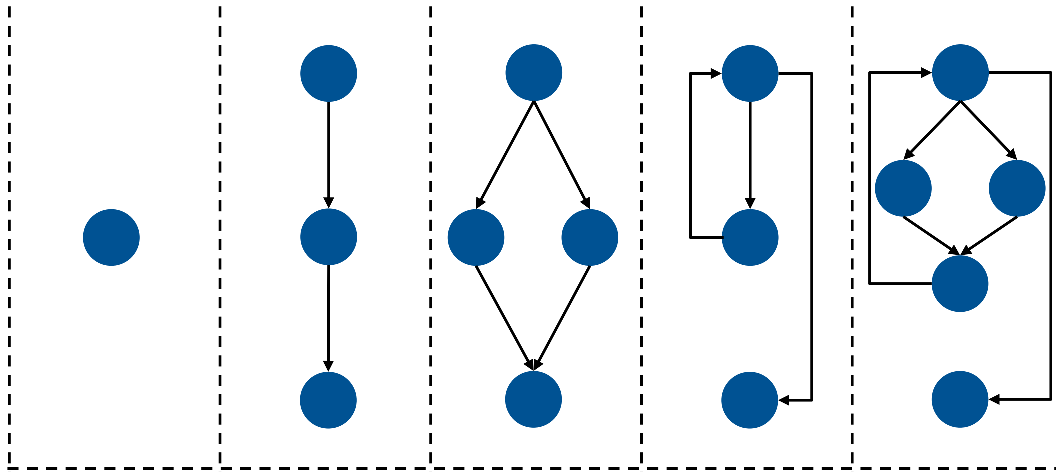
Here, R denotes reusability, L_r is the count of reused lines in the source code, and L_n is the number of newly added lines in the component. A high R value, approaching 1, signifies minimal code adaptation, whereas a value near 0 indicates substantial new line additions. An R value of 0 implies complete reuse without any modifications. This metric, while evaluating reusability through project comparison, does not support static component analysis. It aids in deciding whether integrating new functionalities into an existing component is viable or if creating a new component is more appropriate (Koteska and Velinov, 2013).

Solution-oriented metrics concentrate on the outcomes derived from components. Poulin, Caruso, and Hancock (1993) introduce metrics based on Lines of Code (LoC) to gauge reuse. They propose the Reuse Percent and Adaption Percent metrics, quantifying the ratio of reused and adapted lines in a software solution. Adapted for CBSE in PdM, these metrics focus on pertinent components, circumventing distortions from ancillary code (Poulin, Caruso, and Hancock, 1993).

Cyclomatic complexity, a concept introduced by McCabe (1976), stands as a prominent White-Box metric for the structural analysis of software components via control flow graphs. It is calculated as:

$$V = e - n + 2p \tag{4.3.2}$$

where V is the cyclomatic complexity, e represents the number of edges in the graph, n is the number of nodes, and p is the number of connected components. This metric is used to determine the complexity of a program's flow, with a lower value indicating better understandability and testability due to fewer potential test cases. High cyclomatic complexity often suggests lower code quality. Figure 21 illustrates an example of calculating cyclomatic complexity in a software component.



e (edges)	0	2	4	3	6
n (nodes)	1	3	4	3	5
p (components)	1	1	1	1	1
V (CK-metric)	1	1	2	2	3

Figure 21: Exemplary calculation of cyclomatic complexity in a software component. e: Number of edges in the graph; n: Number of nodes; p: Number of connected components; V: Cyclomatic complexity (own figure).

Summed up, White-Box metrics provide a detailed insight into the internal aspects of software components, focusing on code-based reusability. These metrics, including the amount of reused code, cyclomatic complexity, and solution-based evaluations, are essential for assessing the suitability of components for a PdM component repository. They guide decisions on whether to include, modify, or expand components, ensuring high-quality and efficient reuse in the development process.

Black-Box metrics typically rely on information provided by a component without involving its source code. Washizaki, Yamamoto, and Fukazawa (2003) present a model for software component reusability using Black-Box metrics, focusing on attributes like understandability, adaptability, and portability. Understandability is defined by the estimated effort required for a user to grasp the concept and usability of a component, with the Existance of Meta-Information (EMI) being a key metric (Washizaki, Yamamoto, and Fukazawa, 2003). Figure 22 illustrates these three dimensions of Black-Box metrics—understandability, adaptability, and portability—along with their individual metrics.

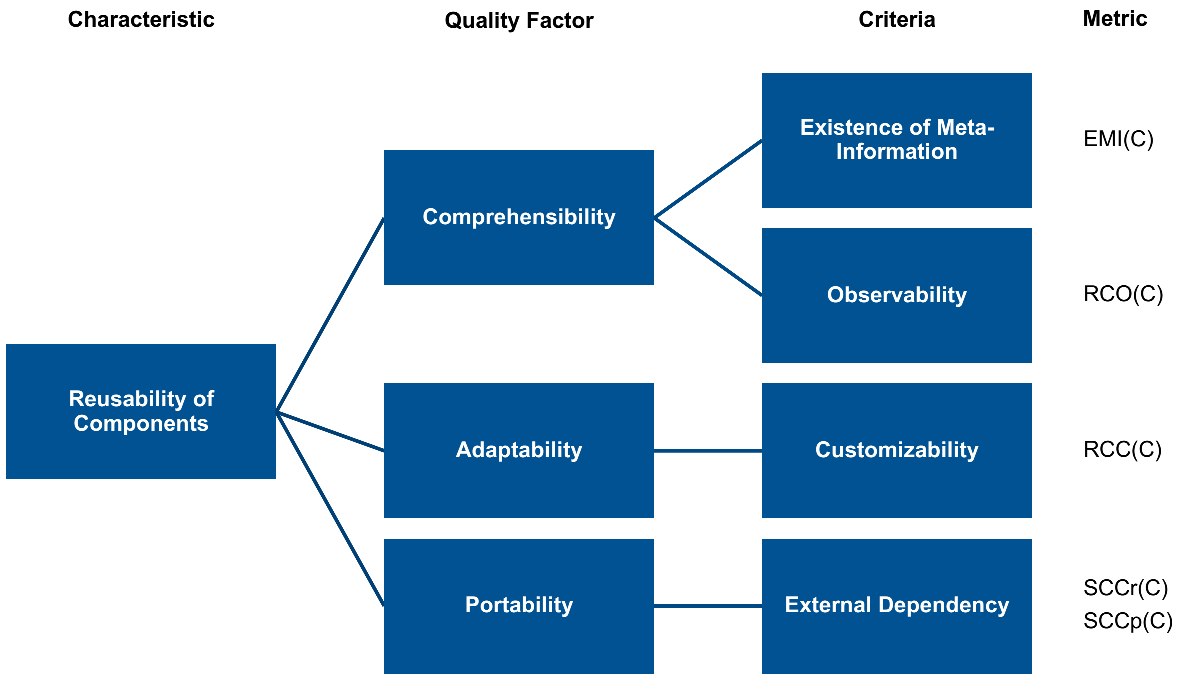


Figure 22: Black box component reusability metrics with the dimensions comprehensibility, adaptability and portability, along with their individual criteria and metrics based on Washizaki, Yamamoto, and Fukazawa (2003).

Understandability is defined by the estimated effort required for a user to comprehend the concept and usability of a component. A central metric for this attribute is the EMI. The EMI is quantified as:

$$\text{EMI} = \begin{cases} 1, & \text{if meta-information exists} \\ 0, & \text{otherwise} \end{cases} \tag{4.3.3}$$

A component is considered more reusable if it achieves an EMI value of 1, indicating the presence of clear, readable, and structured documentation (Washizaki, Yamamoto, and Fukazawa, 2003). **Adaptability** refers to the ease with which a component can be modified to meet new requirements. The Component Adaptability Rate (RCC) is defined as:

$$\text{RCC} = \begin{cases} \frac{P_w(c)}{A(c)}, & \text{if } A(c) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.3.4}$$

where $P_w(c)$ represents the number of write methods in component c, and $A(c)$ denotes the number of writable fields in c. The RCC value indicates the degree of adaptability of the component to the user. An alternative formulation, more suited to the PdM process, is the Component Function Adaptability Rate (RCC_f), given by:

$$\text{RCC\_f} = \begin{cases} \frac{P_h(c)}{P_c(c)}, & \text{if } P_c(c) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.3.5}$$

where $P_h(c)$ is the number of parameters in the function header, and $P_c(c)$ is the total possible parameters in the function. A high adaptability rate is desirable for understanding component behavior, but excessive adaptability might confuse users, leading to misuse or avoidance of the component (Washizaki, Yamamoto, and Fukazawa, 2003).

Portability metrics, such as Self-Completeness-of-Component-Return-Value (SCCr) and Self-Completeness-of-Component-Parameter-Value (SCCp), assess the ease with which a component can be transferred and used in different environments. While these metrics are important, they are less relevant in the PdM context due to the specific application and definition of components in the repository.

Consistent naming of function parameters enhances reusability and quality of the component repository. Boxall and Araban (2004) introduce the Mean String Commonality (MSC_A) metric, adapted here as the Mean Identifier Consistency (MIC):

$$\text{MIC} = \frac{1}{k} \sum_{n=1}^{k} \min_{j \in B} D(A_n, B_j) \tag{4.3.6}$$

where $D(A_n, B_j)$ represents the Levenshtein distance between parameter A_n of the new function and each parameter B_j in the target module, and k is the number of elements in A. A higher MIC value suggests a discrepancy in parameter naming conventions, indicating a need for revision to align with the module's standards (Boxall and Araban, 2004).

Self-documentation is a key feature of high-quality, reusable software components. The Mean Identifier Length (MeIL) metric, proposed by Boxall and Araban (2004), measures the median length of function identifiers. A higher MeIL value implies better self-documentation, but excessively long identifiers can reduce reusability due to less frequent use of overly detailed interfaces.

In summary, the literature on reusability metrics provides a comprehensive framework for evaluating components in a PdM environment. White-Box metrics focus on the internal aspects of components, assessing their code-based reusability, while Black-Box metrics evaluate components based on external factors like understandability and adaptability. These metrics collectively aid in determining the suitability of components for a repository, guiding decisions on whether to include, modify, or expand them. The adaptation of these metrics to the specific needs of PdM development is important for maintaining a high-quality, efficient component repository.

Despite the utility of reusability metrics in evaluating components for a PdM environment, a notable research gap remains in the adaptation of these metrics for the specific needs of PdM development. The literature thoroughly discusses the assessment of components through White-Box and Black-Box metrics, focusing on internal and external qualities. However, it largely overlooks the necessity of tailoring these metrics to guide the design and management of a component-based repository specifically for PdM. This oversight suggests a missing link in the current understanding of how to effectively apply these metrics to enhance the development efficiency of PdM models.

Moreover, the absence of detailed guidelines on incorporating these metrics into the repository design process highlights a critical gap in the literature. For PdM applications, where reliability and prediction accuracy are central, the design of component repositories requires a nuanced approach that accounts for the unique challenges and requirements of PdM systems. This includes considerations for the diverse operational environments in which PdM systems are deployed and the need for components that can seamlessly integrate with ML models to process and analyze varied data types.

To bridge this research gap, there is a pressing need for studies that not only refine existing reusability metrics for PdM but also develop comprehensive frameworks for applying these metrics in the construction and optimization of component repositories. Such frameworks should offer clear, actionable guidelines that are specifically designed to support the development of PdM models, ensuring that components are both reusable and finely tuned to meet the demands of PdM applications.

In this work, the identification and execution of initial and subsequent use cases within a real-world environment represent a crucial step towards addressing the research gaps highlighted previously. These use cases, specifically chosen for their relevance to PdM development, serve as the foundation for a practical exploration into the application of component-based development principles. By implementing these use cases, this work not only tests the applicability of the proposed reusability metrics in a live PdM setting but also allow for the distillation of design rules that are uniquely tailored to the nuances of PdM.

The execution of these use cases in a real-world PdM environment allows for the direct observation and analysis of component-based development in action. This approach enables the identification of key factors that influence the effectiveness of component reusability and the overall efficiency of the development process. By closely examining the outcomes and challenges encountered during these use cases, this work derives a set of design rules based on the interplay between established reusability measurements and the specific requirements of PdM.

These design rules, informed by both the theoretical framework of reusability metrics and the practical insights gained from the use cases, offer a targeted response to the previously identified research gaps. They provide concrete guidelines for the design and management of component repositories in PdM development, ensuring that the components not only meet the general criteria for reusability but are also optimized for the specific demands and challenges of PdM applications. Thus, this work contributes significantly to the field by translating theoretical metrics into actionable strategies for enhancing the development efficiency of PdM models.

To transition the research findings of this work into a practical context, a domain-specific PdM use case within an industrial real-world environment is required. This use case is intended to implement the described approach. Selecting a novel area within the domain is vital, aiming for a subject yet to be extensively explored in PdM. This strategy paves the way for an execution independent from previous practices in the field of PdM.

The second step involves the execution based on the outlined process, aiming to validate whether this process is suitable for achieving the required outcomes of the use case. Additionally, it assesses whether it can help identify components that can be added to the component repository.

In step three, the selection of an additional use case with similar conditions within the PdM domain aids in demonstrating the process's appropriateness and the efficacy of CBSE. This involves comparing various use cases in terms of their required inputs and outputs to choose one as similar as possible to the initially conducted one.

Step four involves the execution of the PdM use case selected in step three, following the introduced process. This step validates whether the process is appropriate for meeting the use case's required outcomes. Additionally, it examines whether this process can identify components that could be added to the component repository.

Step five involves using the metrics found in the literature to formulate a well-founded answer to the research questions, following the implementation and execution of both use cases.

This detailed approach to research execution and evaluation encompasses phases 3 to 6 of the introduced DSR process. The implementation of these steps is described in section 3.1.

The initial use case focuses on the theme of brake noise detection within the context of this work's real-world industrial environment, setting it against the backdrop of the *Predictive Maintenance Framework* introduced in section 2. This use case leverages microphone recordings to develop a ML model capable of identifying unwanted noises in brake systems, thereby predicting potential failures or excessive wear over time. According to the *Predictive Maintenance Framework*, the use case is classified as follows:

The data source for this use case comprises microphone recordings (sensor data) from test drives designed to reproduce noises under various conditions. These recordings are primarily from the vehicle's interior to easily measure noise levels relevant to the driver caused by the brake systems. The high-resolution recordings (48,000Hz) are presented as time series, providing a detailed basis for subsequent analysis.

Feature engineering for this use case involves frequency-based characteristics, where audio recordings, specifically during known intervals of the targeted noises, are segmented into overlapping two-second sections. These segments are then transformed into visual representations through Fast Fourier Transformation and Melody (MEL)-Spectrograms, facilitating their integration into a modeling technique suitable for a Convolutional Neural Network (CNN) architecture. This modeling approach aligns with data-based prediction, specifically supervised learning, as the data labeling is precisely defined by the department responsible for the test drives.

This use case also highlights anomaly detection, aiming to identify noises that deviate from the normal operation of brake systems. Given the high frequency of the data and the extensive duration of the test drives, a significant challenge arises as the collected time series cannot entirely fit into the memory of the available computing resources. This aspect is fundamental for planning the components required for implementation. Additionally, the detailed description of the labels and their timestamps, and the .wav file format of the data, are essential considerations for the component design, indicating whether existing components can accommodate the data integration needs.

In the structure depicted in Figure 9, the code implementation of the initial use case along with the LoC is showcased, aligning with the development process defined in section 2 of this work. Following the problem analysis detailed in the preceding sections, the search for pre-existing components in the component repository is undertaken. Given the intent to implement a novel use case not previously addressed, only one component is found applicable for implementation.

This component's functionality is to converting a given time series into equal-length, overlapping segments regardless of their frequency.

Component Name	**New**	**LoC**
Module: DI		
`read_wav_to_pandas_DF`	`True`	21
Module: DP		
`write_pandas_DF_to_wav`	`True`	14
`create_metadata_DF`	`True`	20
`write_DF_to_csv`	`True`	36
`read_csv_to_DF`	`True`	4
Module: FC		
`get_windows_dict`	`False`	46
Module: FL		
`create_label_for_timestamp`	`True`	12
Module: MB		
`get_soundDS`	`True`	153
`get_train_test_DataLoader`	`True`	11
`get_AudioClassifier`	`True`	80
`train`	`True`	54
`test`	`True`	54
Module: ME		
`get_confusion_matrix_fig`	`True`	78
		583

Table 9: Overview of the required components for the initial use case *High-Freq Brake Noise Detection* with the corresponding modules and the LoC per component.

Subsequently, components required to address the relevant characteristics of this use case are defined. Derived from the limitation that the entire time series cannot be accommodated in the computing machine's memory, a component is needed to keep only the time series required for the current training iteration in memory while storing the remaining samples on disk. To realize intermediate storage in the desired .wav format, another component providing this functionality is essential. Additionally, considering the input in a .wav file format, a component for reading this format into a data structure processable by the ML model is necessary.

Following the problem analysis and planning phase, the implementation of the solution as outlined in the development process proceeds. After implementation, the model is evaluated in collaboration with the department that commissioned the use case, ensuring it meets the set objectives. After fine-tuning the data basis regarding label assignment, the trained CNN achieves specific results on the test dataset (383 samples) after 500 training epochs.

As part of the process, the team collaboratively reviews the developed solution for new components to be added to the component repository. The program's flow from top to bottom, including the names of the used components, is detailed in the corresponding table. The second column indicates whether a component is reused or newly created, while the third column reflects the

LoC metric discussed as a white-box-metric, showcasing the size of functionalities encapsulated within components. In total, 583 LoC are required for the implementation of this industrial use case in the PdM domain, excluding lines needed for component definition and implementation approach (function headers and documentation). Out of these, 38 lines are reused from an existing component, and 525 lines are encapsulated into new components and added to the component repository, following the defined rules. This resulted in the reuse of one component and the creation of 13 new components.

The second use case builds directly upon the initial theme of brake noise detection, introducing a novel aspect within the same real-world industrial *Predictive Maintenance Framework*. This case explores the feasibility of using existing vehicular sensors, specifically wetness sensors in the wheel wells, to classify brake noises into various categories. This approach aims to evaluate the potential for offering additional services to customers through software updates, enhancing the existing product's value with a focus on PdM.

The Communication Bus (C-Bus) logger data (sensor data) is selected from test drives that intentionally replicated braking noises under diverse conditions. The domain knowledge is crucial in selecting signals that likely capture the brake noises. Following a detailed analysis, wetness sensors, recording at a frequency of 40,000Hz, are chosen due to their close approximation to the specifications of the audio recordings discussed in the first use case.

The feature engineering process for this case mirrors the initial use case, focusing on frequency-based characteristics. The sensor recordings, with known intervals from the first use case where the target noises occurred, are segmented into overlapping two-second sections. These segments are scaled to match the length of the audio recordings and then transformed into visual representations through Fourier Transformation and MEL-Spectrograms, making them compatible for modeling with a CNN architecture.

Similar to the first use case, this scenario adopts a data-based prediction strategy, specifically supervised learning, based on precisely defined labels by the department conducting the test drives.

The aim is anomaly detection, identifying noises or areas that represent deviations from the brake system's normal operation, leveraging the unique capabilities of the selected sensors.

The high frequency and extensive duration of the test drives pose a challenge, as the collected time series data cannot fully fit into the available computing resources' memory. This limitation necessitates careful planning for the required components for implementation. Additionally, the precise description of the labels and their timestamps simplifies the process of correctly assigning labels within the data set. The use of .parquet file format for C-Bus recordings indicates the potential for existing components to aid data integration into the workflow.

This second use case not only validates the flexibility and applicability of the developed components in a different context but also contributes to validating the overall development process under the CBSE framework. By testing the components in varied scenarios, this use case reinforces the repository's value, showcasing the adaptability and efficiency of the component-based approach in the PdM domain.

In the execution of the second use case, the development process emphasizes validating components from the initial use case and assessing the CBSE approach for PdM applications. The decision to select a use case that closely mirrors the characteristics of the first use case allows for the substantial reuse of previously developed components. The notable exception is the necessity for a new component to handle the .parquet data format, replacing the .wav file reading component due to the new data specifications. Table 10 shows the components used in the execution of the second use case.

Component Name	**New**	**LoC**
Module: DI		
`read_parquet_to_pandas_DF`	`True`	172
Module: DP		
`write_pandas_DF_to_wav`	`False`	14
`create_metadata_DF`	`False`	20
`write_DF_to_csv`	`False`	36
`read_csv_to_DF`	`False`	4
Module: FC		
`get_windows_dict`	`False`	46
Module: FL		
`create_label_for_timestamp`	`False`	12
Module: MB		
`get_soundDS`	`False`	153
`get_train_test_DataLoader`	`False`	11
`get_AudioClassifier`	`False`	80
`train`	`False`	54
`test`	`False`	54
Module: ME		
`get_confusion_matrix_fig`	`False`	78
		724

Table 10: Overview of the required components for the following use case *Low-Freq Brake Noise Detection.*

A component required for scaling sensor recordings to amplitudes suitable for .wav formats is also identified, enabling the use of sequential data reading components for training iterations. This adaptation signifies the CBSE methodology's capability to efficiently repurpose existing components with minimal modifications.

The collaborative effort results in the implementation and evaluation of the model, which is subjected to multiple training iterations. This process highlights the adaptability of the developed components, demonstrating their effectiveness across different data formats and use case specifics.

The resulting CNN models showcase varying degrees of success across test datasets. The discrepancies in model performance are attributed to challenges related to data restoration and the direct transfer of labels from microphone recordings, underscoring the complex nature of PdM data analysis. The findings from this use case further validate the CBSE approach, illustrating its potential to rationalize PdM development by leveraging reusable components.

A total of 724 LoC are required for the implementation of this second industrial use case in the PdM domain. Of these, 553 lines are reused from existing components, with an additional nine lines encapsulated into new components added to the repository. This reuse and expansion process results in the adaptation of 14 components and the creation of one new component, underscoring the CBSE methodology's efficiency and flexibility in addressing the evolving requirements of PdM applications.

This use case underscores the strategic value of CBSE in PdM, demonstrating the approach's ability to adapt to new challenges while maximizing the reuse of existing solutions.

The culmination of this research, through the diligent application of CBSE principles in the domain of PdM, highlights the method's inherent adaptability and efficiency in addressing complex PdM challenges. By focusing on the development, validation, and adaptation of reusable components across two distinct use cases—brake noise detection using microphone recordings and sensor data from wetness sensors—the research underscores the significant potential of CBSE in streamlining PdM development processes.

This approach not only promotes the rapid assembly and modification of predictive models to meet specific PdM requirements but also demonstrates the value of component reuse and adaptation in enhancing development efficiency. The execution of the use cases serves as a practical test bed for the theoretical concepts discussed, providing concrete evidence of the CBSE methodology's effectiveness in real-world PdM applications.

Moreover, the successful adaptation of components to new data formats and the strategic handling of PdM specific challenges, such as data quality and ME criteria, exemplify the CBSE approach's flexibility. This research contributes to the PdM field by offering a structured framework for component-based development, encouraging a more agile and responsive approach to AI model development in industrial settings.

In conclusion, this work presents a compelling case for the adoption of CBSE in PdM development, highlighting its advantages in component reuse, system adaptability, and development efficiency. The insights gained from the implementation of the described use cases contribute valuable knowledge to the domain, paving the way for future research and application of CBSE in PdM and beyond.

4.4 Workflow for Component Creation and Comparison

CBSE emerges as a vital approach in the development of modular, scalable, and maintainable software systems. This paradigm emphasizes the reuse of existing software components and the creation of new components as fundamental building blocks for software applications. The literature in this domain spans various aspects, including quality assurance, regression testing, business-oriented development, component reuse, model maintenance, and dynamic component integration. This literature review aims to synthesize key contributions from seminal works in the field, focusing particularly on the methodologies and strategies for the creation of new components in CBSE. Examining these studies seeks to understand the evolving landscape of CBSE component creation and its implications for efficient software development and maintenance.

D. Kumar and Kumari (2015) discuss the critical role of quality assurance models and metrics in CBSE. The paper emphasizes the importance of ensuring the reliability and efficiency of software components, which are fundamental in the creation of new components. The study likely explores various quality assurance models and metrics, providing insights into how these can be applied to assess and enhance the quality of new software components.

Orso et al. (2001) explore the use of component meta-content in regression testing. This research is significant for the development of new components as it addresses the challenges in ensuring that changes in components do not adversely affect the existing system. The methodologies and findings from this paper can provide valuable guidelines for testing new components in CBSE.

Jarzabek and Hitz (1998) discuss the alignment of CBSE with business objectives. This paper is crucial for understanding how new components can be developed with a business-centric approach, ensuring that they meet market demands and contribute to the strategic goals of the organization.

Edmunds, Snook, and Walden (2016) investigate into the reuse of components in the context of Event-B, a formal method for system modeling. This paper's focus on component reuse is particularly relevant for the creation of new components, as it provides insights into how existing components can be effectively adapted and integrated into new software solutions.

Jahn et al. (2012) address the challenges in maintaining models in component-based product lines. This research is pertinent to the development of new components as it highlights the importance of maintaining consistency and compatibility within a product line, which is a key consideration when introducing new components.

Finally, Kathirvelkumaran and Moorthy (2023) presents an architectural pattern for integrating components dynamically. This recent study is particularly relevant for the creation of new components, as it explores innovative approaches to component integration, which is a critical aspect of CBSE.

In conclusion, the reviewed literature underscores the significance of component creation in CBSE. The development of new components, guided by robust quality assurance models, regression testing strategies, and business-oriented approaches, is essential for building scalable and efficient software systems. The ability to define components within a useful scope and maintain an overview of existing and available components is essential for ensuring system integrity and alignment with business goals. Studies like those by D. Kumar and Kumari (2015), Orso et al. (2001), Jarzabek and Hitz (1998), Edmunds, Snook, and Walden (2016), Jahn et al. (2012), and Kathirvelkumaran and Moorthy (2023) provide valuable insights into various facets of component creation, from quality assurance and testing to dynamic integration and model maintenance. This focus on component creation is not just a technical necessity but also a strategic approach to ensure that software systems remain adaptable, maintainable, and relevant in a rapidly evolving technological landscape.

4.4.1 Introduction of the Workflow Design

As part of the ongoing exploration into PdM model creation, Figure 23 presents a comprehensive process flow graph. This graph outlines the systematic approach employed in the initial creation of a model, as well as the methodologies applied in subsequent (n+1) model development cycles. The process is thoroughly designed to integrate and leverage the modular components defined in subsection 4.2. This unified workflow encapsulates the entire spectrum of model creation, from conceptualization to deployment, ensuring a cohesive and repeatable process. The subsequent sections will conduct a detailed analysis of this workflow, highlighting its efficiency and effectiveness in fostering robust model development.

The process flow, as illustrated in Figure 23, is structured into three distinct yet interlinked recursive layers: the **Model Layer**, the **Module Layer**, and the **Component Layer**. Each of these layers operates with its dedicated database, ensuring an organized flow of information. The journey through these layers commences from the `Initiate_Model` node, marking the inception of the model creation process. As the workflow progresses, each layer is methodically traversed, culminating at the `Execute_Model` node. This layered architecture not only enhances the clarity and manageability of the model creation process but also ensures that each aspect of the model is thoroughly developed and integrated. The following sections will look into the specifics of each layer, explaining their unique roles and contributions to the overall process.

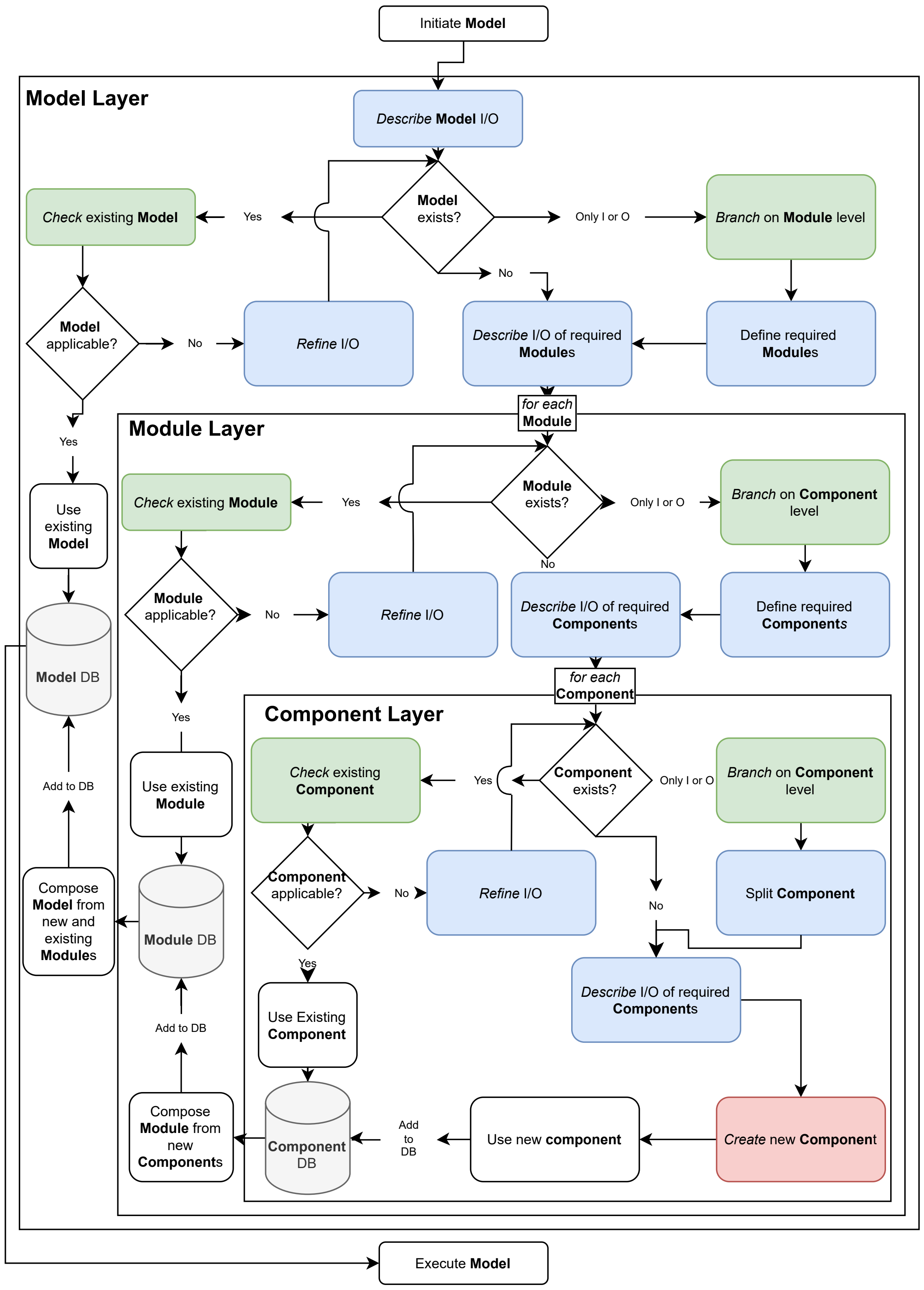


Figure 23: *Component Creation Workflow* as process flow graph, structured into *Model Layer*, *Module Layer* and *Component Layer*. White nodes: Fully automated; Blue nodes: Descriptive efforts required; Red nodes: Coding efforts required (own figure).

The initial phase in the model creation process, as depicted in Figure 23, commences with the **Model Layer**. Upon initiating a model, its inputs and outputs are thoroughly described using a formalized attribute system, detailed in subsection 4.5. This system encompasses all known details of the data inputs available for model development and explicitly defines the model's objectives through output attributes. Subsequently, this model description undergoes a thorough verification against the `Model_DB` database, which archives all pre-existing models.

This verification can result in one of three distinct pathways:

1. If no existing models share similar input or output attributes with the new model, the process advances to the `Describe_Inputs/Outputs_of_required_Modules` node. Here, each required model module is described, again utilizing input and output attributes. The number of modules hinges on the model process exit point, as explained in subsection 4.2, with the defined interfaces between modules informing the attribute description.
2. In cases where either the input or output attributes partially align with an existing model, the developer is prompted to manually inspect the corresponding model. This inspection aims to identify potential for reusing modules from the existing model, leading to the definition and creation of any required new modules in the `Module_Layer`. If there are multiple models that fit the criteria, the developer manually chooses the model that most closely aligns with the requirements to continue the workflow.
3. The third scenario occurs when an existing model fully satisfies the input and output criteria. Here, the developer must analyze the model to ascertain its suitability. If deemed unsuitable, the attributes of both the pre-existing and new models are refined, focusing on demonstrating the reasons for incompatibility. Conversely, if the existing model is suitable, it is executed, thereby concluding the workflow.

In the `Module_Layer`, each module defined in the `Model_Layer` undergoes a critical evaluation process. This evaluation involves checking the described module against the `Module_DB` database to ascertain if similarly described modules already exist. This check can lead to one of three possible outcomes:

1. If no existing module in the `Module_DB` shares similar input or output attributes, a new module is deemed necessary. At this juncture, the developer faces a choice: to construct the module using either a single component or multiple components. If multiple components are selected, each component is individually described using the formalized attributes system for inputs and outputs. Conversely, if a single component is chosen, the input/output

description of the module is directly utilized for component creation.

2. In scenarios where either the input or output attributes partially align with an existing module, the developer is tasked with manually inspecting the proposed module. This inspection may lead to the division of the module at a specific branch point, effectively splitting it into two components. The reusable component, whether it be the first or second part of the split, is then utilized, and the newly required component is described and forwarded for component creation.
3. The third possibility arises when a pre-existing module closely matches the required module in terms of descriptive attributes. Similar to the approach in the `Model_Layer`, the developer is required to manually inspect the recommended module to determine its suitability. If the existing module meets the requirements, it is integrated into the initiated model. If it does not, the descriptive attributes of both the new and pre-existing modules are refined to highlight the causes of incompatibility.

The `Component_Layer` represents the final stage in the model creation process. In this layer, the attributes of the required components are thoroughly compared with those of pre-existing components in the `Component_DB` database. This comparison leads to one of three potential outcomes:

1. If no existing component in the `Component_DB` matches the required input or output attributes, a new component is created. This step involves a detailed description of the new component using the established attributes system.
2. In cases where either the input or output attributes align with an existing component, a unique approach is adopted. Unlike in the previous layers, the existing component is split into two at a manually determined branch point. This process not only involves the description of the newly required component but also the re-description of the split pre-existing component. Subsequently, the original component in the `Component_DB` is replaced by these two new components, which are positioned before and after the splitting branch point. Both components are then added to the database, reflecting the updated structure.
3. The third scenario occurs when a pre-existing component in the `Component_DB` fully satisfies the required attributes. In this case, the existing component is directly utilized in the model, streamlining the development process.

This approach in the `Component_Layer` ensures a thorough and precise evaluation of components, helping a dynamic and adaptable model creation process. The ability to split and redefine components enhances the flexibility and granularity of the model, allowing for a more tailored and efficient workflow.

Upon the successful creation of each required component in the `Component_Layer`, the process flow advances towards its culmination. The individually crafted components are methodically composed to form their respective modules. These newly assembled modules, along with any pre-existing modules that are utilized or modified, are then systematically added to the `Module_DB` database. This step ensures that all module data is up-to-date and accurately reflects the current state of the system.

Following the module composition, these modules are integrated to construct a new model. This integration is a critical phase where the synergy of the modules is harnessed to meet the predefined objectives of the model. Once this integration is satisfactorily completed, the newly developed model is added to the `Model_DB` database. This addition serves not only as a record of the model's existence but also as a repository for future reference and potential reuse.

The final step in this comprehensive process is the execution of the new model. This execution is the practical application of the entire development effort, where the model is put into operation to fulfill its intended purpose. The successful execution of the model signifies the completion of the process flow, marking the achievement of a significant milestone in the model development journey.

In the context of CBSE, the Process Flow diagram outlines a stark contrast to the monolithic approach, particularly in terms of automation, manual effort, and the requisite skill set at each node. Unlike the monolithic paradigm, where the process is largely linear with minimal nodes between `Initiate_Model` and `Execute_Model`, CBSE introduces a multi-faceted workflow. Each node in this workflow represents a unique blend of automated processes and manual intervention, demanding specific skills and expertise. This section aims to dissect these nodes, offering a comprehensive understanding of the level of automation and manual effort involved. The analysis highlights how CBSE permits a more granular and specialized approach to software development, contrasting it with the more structured but less flexible monolithic model. The discussion serves to underscore the trade-offs and synergies between automated processes and human expertise in the realm of CBSE, setting the stage for a deeper exploration of the individual nodes in the subsequent sections.

The development effort required for a PdM model is significantly influenced by the complexity of the analysis and the quality of the data. While the variance in complexity and quality undoubtedly affects the comparison between development workflows, it is challenging to quantify these effects. For the purpose of this analysis, it is assumed that these influencing factors impact both compared approaches equally. The nodes in the workflow are categorized into four distinct types of developer actions, each represented by a different color.

White nodes are fully automated, requiring no direct action from the developer. These nodes represent processes where the system autonomously performs tasks without human intervention.

Green nodes signify code reading and understanding actions. The effort required here is contingent upon the complexity of the code being analyzed. For instance, analyzing an entire model demands significantly more effort than understanding an individual component. Larger components are generally more challenging to comprehend than smaller ones, assuming no additional information is provided.

Blue nodes involve descriptive efforts. Developers are tasked with articulating the required outputs of the focused functional element and the available inputs. The effort in this stage is influenced not only by the complexity and data quality but also by the description environment and the flexibility of the functional element.

Red nodes, finally, represent actual coding efforts. In the workflow diagram, this is exemplified by the `Create_new_Component` node. Here, developers are required to develop a component that cannot be substituted by existing components or a combination thereof. This node demands a high level of coding expertise and creative problem-solving skills.

In contrast to the nuanced and specialized approach of CBSE, a monolithic approach requires the involvement of every type of developer activity identified in the workflow model. This encompasses the full spectrum of actions, from fully automated tasks (white nodes) to intensive coding efforts (red nodes). Such a comprehensive requirement of diverse skills and efforts in a monolithic system underscores the importance of thoroughly analyzing individual process paths. This detailed examination is key to effectively compare the workflow model of CBSE with the monolithic benchmark. It allows for a clearer understanding of the efficiencies and challenges inherent in each approach, highlighting the areas where CBSE can offer significant advantages in terms of specialization and modularity. This comparison not only sheds light on the operational differences but also provides insights into the strategic implications of choosing one approach over the other in the context of software development and maintenance.

The presented development workflow will be characterized by three distinct paths, each representing a different stage in the evolution of the workflow.

Model 0 - Initial Workflow Initiation: The first path, referred to as Model 0, marks the commencement of the workflow. This stage is characterized by the absence of pre-existing models, modules, or components. Developers are required to initiate the workflow from scratch, which involves a significant amount of coding and design effort, as there are no existing resources to leverage.

Early Stage Workflow Establishment: The second path represents an early stage of workflow establishment. At this juncture, some models and components are already in place. However, the number of models is still manageable enough to allow for time-efficient analysis. The descriptions of functional elements at this stage are not yet refined in detail, necessitating a balanced mix of coding, descriptive, and analytical efforts.

Far Established Scenario: The final path describes a far established scenario. In this advanced stage, the sheer number of models renders individual analysis inefficient. A substantial repository of components exists, and the descriptions of functional elements are refined to a detailed level. This stage emphasizes the importance of efficient component reuse and integration, with a reduced need for new coding efforts. The workflow in this scenario is highly automated, with a focus on maintaining and refining the existing component base.

These paths illustrate the evolving nature of the workflow in CBSE, highlighting the shift from intensive development in the initial stages to a more maintenance and refinement-oriented approach in the later stages.

In the **Model 0** path of the workflow, the initiation of the model and the description of resources (Input) and requirements (Output) align closely with the monolithic benchmark. The absence of comparable models in this initial stage requires that the developer describes each module individually, focusing on input and output attributes. Given that most modules are sequential, the interface between two modules is defined by the output of the preceding module and the input of the subsequent one. The number of descriptive interfaces in this scenario is given by $(n_{\text{modules}} + 1) - 2$, considering that the model's input and output serve as interfaces to the first and last modules, respectively. An additional descriptive effort is required for non-sequential modules, such as Utilities and Data Analytics, where the module's output must be explicitly defined resulting in a maximum of $+2$. This represents an added workload compared to the monolithic approach, influenced linearly by the efficiency of the infrastructure for describing attributes.

In the absence of existing modules, developers have the option to divide each module into multiple components. For comparison purposes, it is assumed that modules are not subdivided, leading to the automatic generation of one component per module, with input and output attributes inherited from the module interfaces. The decision to split modules into components, driven by the need to formalize intermediate functional steps or manage module complexity, incurs additional effort. However, it is assumed that such an optional decision is motivated by similar considerations in a monolithic approach, thus not significantly impacting the comparative analysis.

The final step in Model 0 involves the creation of individual components. The predefined input and output descriptions provide a clear task framework for this process. As discussed in section 4.2, the insights from modularization indicate that the functional division into modules is also a common practice in monolithic approaches. Therefore, there is no discernible loss in efficiency in this aspect of the workflow approach compared to the monolithic benchmark.

During the **Early Stage Workflow Establishment**, developers encounter three primary scenarios at the model layer, each presenting unique comparative considerations against the monolithic benchmark.

Scenario 1 - No Existing Similar Model: If no existing model closely matches the current requirements, the additional efforts align with those described in the Model 0 comparison. The development process mirrors that of initiating a new model from scratch, as in the monolithic approach.

Scenario 2 - Existing Model with Same Description: When an existing model with a matching description is identified, the first step involves analyzing its applicability. If the model is deemed suitable, the process concludes here, contrasting with the monolithic approach where developing a new model from the ground up is necessary. The effort saved in development is offset by the effort spent in analyzing the reusable model. However, as the number of models increases, so does the likelihood of finding a reusable model, even with the added complexity of potentially having to analyze multiple models to find a fit.

Scenario 3 - Partial Attribute Match: In cases where only one set of attributes (input or output) matches the current model, an analysis is required to determine the feasibility of reusing parts of the model. This analysis, while adding extra effort, may lead to significant savings in development time for the reusable modules. The possibility of non-compatibility, calling for refinement of the model description, is also considered, mirroring the efforts required in Scenario

2.

This process is recursively applied at the module and component layers. The presence of existing components introduces additional analysis efforts, balanced against the potential savings from reusing components. As the number of modules grows, the granularity of components increases, affecting both the manual analysis effort for reusability and the likelihood of finding suitable components. The overall requirement for new components remains at least on par with the Model 0 scenario, with additional efforts encompassing potential analysis and refinement of functional elements at every layer.

As the workflow progresses, there is a notable increase in the number of potentially reusable functional elements, accompanied by a refinement in the description of their inputs and outputs. This refinement plays a vital role in enhancing the accuracy of matching functional elements to current requirements. Consequently, while the initial stages may see a higher incidence of wrongfully matching elements due to less precise descriptions, this issue is expected to level out over the entire lifetime of the workflow. The ongoing process thus not only expands the repository of reusable components but also improves the precision with which these components can be matched to new development needs. This dynamic is a key factor in the evolving efficiency of the workflow approach, as it gradually converges towards a state where the effort involved in identifying and reusing existing components becomes more predictable and organized, offsetting the initial investment in component analysis and refinement.

In the **Far Established Scenario** of the workflow, developers encounter a landscape rich with existing models and modules. This abundance makes it increasingly rare to find scenarios where neither inputs nor outputs are previously defined, almost guaranteeing a reduction in the need for developing new components. The primary effort in this mature stage shifts towards the analysis and refinement of existing functional elements.

With a high number of elements and a concurrently growing level of detail in their descriptions, the efforts required for effective comparison and detailed articulation of non-compatibility reasons intensify. This calls for a structured approach to the attribute system, as outlined in section 4.5. The detailed and precise descriptions aid in accurately identifying the suitability of existing components for new requirements, thereby streamlining the development process. However, this also means that a significant portion of the developer's time and resources is dedicated to navigating and understanding the extensive component repository. The Far Established Scenario thus represents a phase where the efficiency gains in component development are counterbalanced by the increased complexity in component analysis and selection, underscoring the importance of a well-organized and clearly defined attribute system in managing the workflow at this advanced stage.

4.4.2 Structure of the Discrete Event Simulation Model

In the pursuit of optimizing the development of analytics models, particularly in the context of reusability, this work confronts a significant challenge: the developing stage of the proposed process model does not yet amass a representative volume of empirical data. This limitation is not uncommon in innovative domains, where new methodologies outpace the accumulation of extensive real-world data. To address this challenge, an event-discrete simulation model is constructed. The rationale for employing a DES approach lies in its quality at modeling complex systems with event-driven dynamics, a characteristic intrinsic to the process of analytics model development (Greasley and Edwards, 2021; Y. M. Lee et al., 2007). This simulation serves a dual purpose: firstly, to validate the proposed process model in a controlled, yet realistic virtual environment, and secondly, to derive a cost function for reusability. The latter is particularly crucial as it addresses the third research question of this work, focusing on the economic aspects of reusability in analytics model development. By simulating various scenarios and configurations within the DES framework, and employing state-of-the-art sensitivity analysis techniques (Sfeir, Antoniou, and Abbas, 2018), the costs associated with different levels of reusability can be extrapolated and analyzed . This provides valuable insights even in the absence of extensive empirical data (Çetinkaya, Verbraeck, and Seck, 2015; Hasan, Bahalkeh, and Yih, 2020).

The development of this works event discrete simulation model is central in understanding and optimizing the workflow of analytics model development. DES is particularly suited for this purpose due to its ability to model complex systems where changes and events occur at discrete points in time. This characteristic is fundamental in analytics, where processes and data flows are often event-driven and subject to variability (Byrum et al., 2016; Lopinski, Sachweh, and Müller, 2023). The DES model captures the sequential and conditional processes inherent in analytics model development, including data collection, preprocessing, model training, evaluation, and deployment. This approach allows for a detailed examination of each stage in the analytics workflow, enabling the identification and analysis of bottlenecks, resource utilization, and process efficiencies (Greasley and Edwards, 2021; Y. M. Lee et al., 2007). By simulating different scenarios and configurations, the model provides insights into the optimal strategies for analytics model development, thereby enhancing decision-making and resource allocation (Çetinkaya, Verbraeck, and Seck, 2015; Hasan, Bahalkeh, and Yih, 2020). Furthermore, the integration of DES with data analytics techniques, such as ML and big data processing, offers a robust framework for predictive and prescriptive analytics, aligning with contemporary trends in smart manufacturing and data-driven decision-making (Fakhimi et al., 2014; Nonaka et al., 2015).

Technologies and Methods for Discrete Event Simulation Models

DES can be implemented utilizing a diverse array of technologies and methods, each endowed with distinct features and capabilities:

- **General-Purpose Programming Languages:** Languages like Python, Java, and C++ can be used for DES. They offer flexibility and control but require more effort in terms of simulation framework development.
- **Specialized Simulation Software:** Tools like Arena, Simul8, and FlexSim are designed specifically for simulation. They provide built-in features for modeling, animation, and analysis but can be less flexible for highly customized simulations.
- **Simulation Libraries for Programming Languages:** Libraries like *SimPy* (Python), SimJava (Java), and Simulink (MATLAB) offer a balance between flexibility and ease of use. They extend the capabilities of programming languages with simulation-specific functionalities.
- **Agent-Based Modeling Platforms:** Software like NetLogo and AnyLogic support DES as part of broader agent-based modeling capabilities. They are useful for simulations involving complex interactions and behaviors.

For this project, Python's *SimPy* library is chosen to develop the DES. *SimPy* is an open-source, process-based discrete-event simulation framework. The choice of *SimPy* is motivated by Python's ease of use, readability, and the extensive ecosystem of libraries available for data analysis and visualization.

1. **Defining the Environment:** The first step in using *SimPy* is to create a simulation environment. This environment acts as a container for the simulation processes and manages the simulation time.
2. **Creating Processes:** In *SimPy*, the dynamics of the DES are defined through processes. These are Python generator functions that model the behavior of the components in the simulation, such as machines in a factory or clients in a queue.

3. **Using Events:** The simulation progresses through events. *SimPy* allows the scheduling of events at specific times, triggering changes in the state of the simulation, such as starting or finishing a task.
4. **Running the Simulation:** After setting up the environment and processes, the simulation is executed over a defined period. During this time, *SimPy* processes the events and updates the system state accordingly.
5. **Data Collection and Analysis:** Throughout the simulation, data about the system's performance and behavior can be collected. This data is then analyzed to derive insights, such as identifying bottlenecks or evaluating the impact of different parameters.

The choice of *SimPy* for the development of the DES model offers several advantages. Firstly, the flexibility of *SimPy* as a Python library allows for high customization and seamless integration with other Python libraries, which are essential for data analysis, ML, and visualization. This integration is crucial in developing a comprehensive simulation model that can handle complex analytics tasks. Secondly, the ease of use is a significant factor; Python's syntax, combined with *SimPy*'s straightforward framework, makes the development process accessible not only to experienced developers but also to researchers and practitioners with limited programming background. This accessibility is vital for interdisciplinary research and applications where users may not have extensive coding expertise. Lastly, the robust community and resources surrounding Python and *SimPy* greatly aid in the development and troubleshooting process. The extensive documentation, community forums, and available tutorials ensure that developers and researchers can find the support and information they need to effectively utilize *SimPy* for their simulation needs.

Choice and Function of Model Input Parameters

In DES models, input parameters are categorized based on their behavior or changes over the course of the simulation, primarily into static and dynamic categories.

Static Parameters: Static parameters remain constant throughout the simulation. They are predefined and do not change in response to the simulation's events. These parameters typically represent fixed characteristics of the system, such as the capacity of a machine in a manufacturing simulation or the layout of a facility. They provide a stable framework for the simulation, ensuring consistency in the underlying conditions.

Dynamic Parameters: Dynamic parameters, in contrast, are characterized by their variability and potential to change during the simulation. These changes are often governed by statistical distributions, making them probabilistic in nature. This allows the simulation to incorporate real-world unpredictability and variability. For example, in a healthcare simulation, patient arrival rates might be modeled as a dynamic parameter, fluctuating according to a Poisson distribution. Such parameters introduce randomness and complexity, enabling the simulation to capture the stochastic nature of real-world systems more accurately.

The interplay between static and dynamic parameters is crucial in a DES model. Static parameters provide a consistent baseline, while dynamic parameters introduce the necessary variability and randomness that reflect real-world scenarios' unpredictability and complexity.

Static Parameters

The static parameters in the DES model are defined with specific ranges, reflecting the variability and adaptability of the model to different scenarios. These ranges are fundamental for conducting sensitivity analyses and understanding the impact of parameter variations on the simulation outcomes. All model parameters are displayed in table 11.

Parameter	Lower Range	Upper Range
`BASE_TIME_BUILD_MODEL_MODULE`	10	
`REFINEMENT_SCALE`	5	10
`REFINEMENT_BASE`	1	3
`PARTIAL_SCALE`	2	4
`COMPLEXITY_BRANCH_FACTOR`	0.25	0.50
`BASE_TIME_CHECK_MODEL`	4	16
`BASE_TIME_DESCRIBE_MODEL_IO`	1	10
`BASE_TIME_REFINE_MODEL_IO`	1	15
`BASE_TIME_FIND_MODEL_BRANCH`	4	16
`BASE_PROP_MODEL_USABLE`	1/250	1/100
`BASE_TIME_CHECK_MODULE`	1	8
`BASE_TIME_DESCRIBE_MODULE_IO`	1	10
`BASE_TIME_REFINE_MODULE_IO`	1	15
`BASE_TIME_FIND_MODULE_BRANCH`	4	20
`BASE_PROP_MODULE_USABLE`	1/1000	1/200
`BASE_COMPLEXITY_MODULE`	1	3
`BASE_TIME_CHECK_COMPONENT`	1	8
`BASE_TIME_DESCRIBE_COMPONENT_IO`	1	8
`BASE_TIME_REFINE_COMPONENT_IO`	1	8
`BASE_TIME_FIND_COMPONENT_BRANCH`	2	16
`BASE_TIME_CREATE_COMPONENT`	5	20
`BASE_PROP_COMPONENT_USABLE`	1/1000	1/200
`BASE_COMPLEXITY_COMPONENT`	1.0	1.2

Table 11: Range of static parameters in the DES model with their lower and upper range.

- **Benchmark**:

The benchmark parameter in the DES model, denoted as `BASE_TIME_BUILD_MODEL_MODULE`, is set to a value of 10. This parameter represents the time units required to build an individual model using a traditional, monolithic approach. The choice of the value 10 is strategic, as it allows the comparison of other parameters in the model as integers. For instance, a model requiring 6 modules in the framework of this work would equivalently require 6×10 time units to be constructed in a monolithic manner. This benchmark parameter serves as a constant reference point throughout the simulation, providing a basis for evaluating the efficiency and time-effectiveness of alternative approaches. Unlike other parameters in the model, `BASE_TIME_BUILD_MODEL_MODULE` remains fixed at its initial level during the sensitivity analysis, ensuring a consistent standard for comparison across different simulation scenarios.

It is important to note that these time units are relative and do not correspond to any physically measurable time units. They are used as a standard of comparison within the simulation, providing a consistent metric for evaluating different processes and scenarios. The benchmark parameter serves as a constant reference point throughout the simulation, offering a basis for assessing the efficiency and time-effectiveness of alternative approaches. Unlike other parameters in the model, `BASE_TIME_BUILD_MODEL_MODULE` remains fixed at its initial level during the sensitivity analysis, ensuring a consistent standard for comparison across different simulation scenarios.

- **General Parameters**:

General parameters within the DES model are defined as parameters that are not specific to any individual object layer but rather influence all object layers uniformly. These parameters are integral to the model's framework, ensuring a consistent approach to the enhancement and refinement across different layers. Their uniform application across all layers underscores their foundational role in the overall behavior and usefulness of the DES model, affecting the scalability and adaptability of the system in diverse operational contexts.

`REFINEMENT_SCALE`:

This parameter significantly dictates the level of enhancement achieved in the description of an object at each refinement step. A refinement scale setting of `REFINEMENT_SCALE` = 5 necessitates twenty refinement processes to reach a specified description level, whereas a setting of `REFINEMENT_SCALE` = 10 reduces this requirement to ten steps. Empirical evidence from real-world applications demonstrates that ten refinement steps elevate the description level to a threshold where mismatches in object usage become highly improbable.

`REFINEMENT_BASE`:

This parameter defines the expected base refinement level, vital for establishing the initial level of object description. While observations indicate that the baseline description niveau is inherently high, it is sensible for the workflow model to adopt a conservative approach. Consequently, the `REFINEMENT_BASE` is set to assume a baseline three times lower than what is empirically observed. This cautious assumption ensures that the model remains robust across varying conditions and mitigates the risk of overestimating the initial description capabilities, which could lead to discrepancies in later stages of object refinement and usage.

`PARTIAL_SCALE`:

This parameter denotes the probability of partial applicability of an object. This parameter, `PARTIAL_SCALE`, quantifies the likelihood that an object can be utilized only partially, typically when either inputs or outputs align with the specified description. The lower range value of 2 signifies a doubling of the probability of such partial usability, reflecting scenarios where partial matches between the object's capabilities and the required specifications occur. Conversely, the upper range of 4 indicates a quadrupling of this probability, encapsulating a broader spectrum of partial applicability under more diverse operational conditions.

`COMPLEXITY_BRANCH_FACTOR`:

This parameter quantifies the reduction in complexity achieved through each branching process within the system. Each branch process splits one object into two distinct entities, inherently reducing the complexity associated with each resulting object. The parameter `COMPLEXITY_BRANCH_FACTOR` is critical for understanding this division; a value of 0.5 indicates a straightforward halving of complexity, representing an intuitive decrease. Conversely, a lower limit set at 0.25 suggests a more modest complexity reduction of 25%. This lower range accounts for the potential complexities that branching processes may introduce, acknowledging that new complexities can emerge even as overall complexity per object diminishes.

In the DES model, the processes across the three layers - models, modules, and components - exhibit a fundamental similarity in their nature. However, the scope of these processes varies significantly depending on the layer they relate to. This distinction in scope is key for accurately simulating the different levels of abstraction and granularity represented by each layer.

The range for each parameter in the model is determined based on empirical observations and theoretical considerations. The lower bound of these ranges is derived from the best-case scenarios measured in the real-world environment of this work. These represent the most efficient and structured instances of the processes. Conversely, the upper bound extends to a worst-case scenario, which, while not observed, represents a plausible extreme that could potentially occur. This approach to defining parameter ranges ensures that the simulation encompasses a realistic spectrum of possibilities, from the most favorable to the most challenging scenarios. It allows the model to capture not only the typical behavior of the system but also its response under stress or atypical conditions.

- **Model Layer**:

`BASE_TIME_CHECK_MODEL`:

This parameter represents the time units required to check if a proposed model fulfills the specified requirements. The process involves understanding the model, its required inputs, and its actual outputs. Factors such as the experience of the checking expert and the readability and structuredness of the code are influential but are assumed to be constant for the purpose of this simulation. This assumption is based on the understanding that these factors affect both the benchmark scenario and the workflow scenario equally. The range of this parameter extends from 4 time units, which corresponds to a straightforward check for validating or invalidating the applicability of the proposed model, to 16 time units. The upper limit of this range represents a more detailed analysis of the model results, which is four times more extensive than the best-case scenario and 60% more extensive than the actual creation of one of the model modules. This range reflects the variability in the complexity and depth of the model checking process, accommodating scenarios from quick assessments to in-depth analyses.

`BASE_TIME_DESCRIBE_MODEL_IO`:

This parameter quantifies the time required to describe the inputs and outputs of a model. The process is facilitated by the use case description framework and the attributes system, as introduced in Sections 4.1 and 4.5, respectively. Describing the model I/O is a critical step to ensure comparability with existing models, significantly influencing the entire subsequent process. A thorough and detailed description can lead to more effective results in identifying reusable models but may also reduce the likelihood of finding closely similar models for partial reuse. Conversely, a brief description reduces the initial effort but may increase the time needed for refinement and model checking. The range for this parameter is estimated between 1 time unit, representing a low-effort, high-level description that acknowledges the practitioner's need for efficient time management, and 10 time units, indicative of an exhaustive description process. This extensive process might involve multiple refinement loops with stakeholders and domain experts. The upper bound of this range, being ten times higher than the lower bound, equates to the effort of creating one module-equivalent in a monolithic benchmark process, accounting for approximately 1/5 to 1/10 of the total benchmark process. This range reflects the trade-off between the depth of the I/O description and the overall efficiency of the model development process.

`BASE_TIME_REFINE_MODEL_IO`:
This parameter represents the base time required to refine the model's inputs and outputs in cases where an initial compatibility check results in incompatibility. The refinement process involves specifying the input and/or output attributes in accordance with the parent/sibling structure, as detailed in Section 4.5. This task demands a comprehensive understanding of the model's full functionality, often involving an extensive analysis. In practical scenarios, however, the differences between models might be more readily identifiable. For instance, defining model input attributes can be straightforward if the data inputs are already known. Similarly, the variety of model output attributes is somewhat limited, as evidenced by the analysis in related literature (Section 2). The range for this parameter is set between 1 time unit, reflecting the often-observed duration in real-world environments, and 15 time units, which accounts for a theoretically possible in-depth analysis and understanding of the model's Dp and outcomes. This range captures the spectrum from quick attribute adjustments to comprehensive refinements, accommodating both typical and more complex refinement scenarios.

`BASE_TIME_FIND_MODEL_BRANCH`:
This parameter quantifies the time units required to identify a branch in a model when only parts of the model are reusable, either due to matching input or output attributes. The process calls for a thorough understanding of the model's modular structure and the functionality of its individual modules to determine which modules are applicable for reuse. In real-world scenarios, identifying a suitable branch position is often complex, as there are typically multiple branching options, each with varying requirements for additional modules. Furthermore, the effort involved in this process is not linearly estimatable since modules differ in their development complexity. The parameter encompasses not only the time taken to identify potential branching points but also the time required for decision-making based on the practitioner's estimations. The range for this parameter is set between 4 and 16 time units. This range is based on the resources estimated from the `BASE_TIME_CHECK_MODEL` parameter, reflecting the similarity in the effort and skills required both to understand the model comprehensively and to identify areas of partial reusability. The upper limit of this range accounts for scenarios where in-depth analysis and

decision-making are necessary, while the lower limit represents more straightforward cases of branch identification.

`BASE_PROP_MODEL_USABLE`:

This parameter signifies the probability of finding an existing model that is suitable for a given application. The likelihood of such a match is influenced by factors such as the specific field and industry, the complexity of the input data, and the variance in data types. Given its critical role in measuring the deviation from the benchmark performance (which this parameter does not affect), the range of `BASE_PROP_MODEL_USABLE` is set with careful consideration.

In the real-world environment of this work, it is hypothesized that approximately 50 different models suffice to cover all available data kinds and application fields, setting the base probability at 1/50. To ensure the generalizability of the simulation results to industrial contexts with less variety, the upper bound of the range is set at a reuse probability of 1/100. Conversely, acknowledging that other application fields might encompass a broader array of models, the lower limit is extended to 1/250. This makes `BASE_PROP_MODEL_USABLE` the parameter with the widest range in the model layer, allowing the simulation to explore scenarios from highly specialized to extremely diverse model applicability.

- **Module Layer**

`BASE_TIME_CHECK_MODULE`:

This parameter, with values ranging from 1 to 8, specifies the estimated average time required for checking a module. The range reflects different estimates, accommodating the variability in time needed to validate a module's functionality and compatibility due to differing complexities and sizes of modules depending on industrial contexts.

`BASE_TIME_DESCRIBE_MODULE_IO`:

This parameter, varying from 1 to 10, represents the estimated average time required to describe a module's input and output. The range accounts for different estimates based on the complexity and detail of module interfaces, which can vary significantly across different industrial contexts.

`BASE_TIME_REFINE_MODULE_IO`:

Ranging from 1 to 15, this parameter indicates the estimated average time needed for refining a module's I/O. The broad range reflects the diversity in estimates, accommodating both minor adjustments and more extensive overhauls, which are influenced by the specific requirements and complexities of modules in various industrial settings.

`BASE_TIME_FIND_MODULE_BRANCH`:

With a range from 4 to 20, this parameter specifies the estimated average time required to identify suitable branching points within a module. The range caters to different estimates, reflecting the varying complexities and decision-making processes involved in module integration or modification across diverse industrial applications.

`BASE_PROP_MODULE_USABLE`:

This parameter, varying from 1/200 to 1/1.000, represents the estimated probability of a module being usable. The range encompasses different estimates of usability likelihood, acknowledging the spectrum of module applicability from common to rare in various industrial contexts.

`BASE_COMPLEXITY_MODULE`:

Capturing the estimated average complexity level of a module, this parameter ranges from 1 to 3. The range allows for different estimates of module complexity, accommodating the variability in module design and functionality encountered in different industrial sectors.

- **Component Layer**

`BASE_TIME_CHECK_COMPONENT`:

This parameter defines the range of time required for checking a component, with values from 1 to 8. This range allows for variability in the time needed to assess a component's functionality and compatibility, accommodating components of varying complexity and detail.

`BASE_TIME_DESCRIBE_COMPONENT_IO`:

The time required to describe a component's input and output is represented by this parameter, which varies from 1 to 8. This range reflects the differences in the complexity and detail of component interfaces, from simple to highly sophisticated I/O descriptions.

`BASE_TIME_REFINE_COMPONENT_IO`:

Ranging from 1 to 8, this parameter covers the time needed for refining a component's I/O. The range caters to both minor adjustments and more significant modifications in component I/O refinement.

`BASE_TIME_FIND_COMPONENT_BRANCH`:

This parameter, with a range from 2 to 16, indicates the time required to identify branching points within a component. The range accommodates the varying complexities in determining suitable branches for component integration or modification.

`BASE_TIME_CREATE_COMPONENT`:

Representing the time required to create a component, this parameter varies from 5 to 20. It reflects the diversity in the effort and time needed to develop components, from relatively simple to more complex creations.

`BASE_PROP_COMPONENT_USABLE`:

This parameter signifies the probability of a component being usable, ranging from 1/1.000 to 1/200. It captures the likelihood of finding an existing component that meets specific requirements, spanning a spectrum from commonly usable to rare components.

`BASE_COMPLEXITY_COMPONENT`:

The complexity level of a component is captured by this parameter, which ranges from 1.0 to 1.2. This range allows the model to simulate components of varying complexity, from straightforward to highly complex.

These ranges allow the model to be tested under various conditions, providing insights into how changes in these parameters affect the overall behavior and outcomes of the simulation. This flexibility is essential for a robust and comprehensive analysis, especially in scenarios where the model needs to adapt to different operational contexts or assumptions.

Dynamic Parameters

The simulation model incorporates several dynamic parameters to simulate the complexity and variability of objects within the system. These parameters are crucial for determining the probabilistic existence, applicability, and the required number of objects for branches, which collectively contribute to the model's realistic representation of complex systems.

- **Probability of Object Existence**

The probability of an existing object within the simulation is calculated using a function that integrates a reciprocal function with a logistic function. The logistic curve is defined by Equation 4.4.1 for an object of type "Model", "Module", or "Component".

$$P(x) = \begin{cases} 0 & \text{if } x \leq 1, \\ \left(3x^{-1} + \frac{L_{existing}}{1+e^{-k_{existing}(x-n)}}\right) & \text{otherwise,} \end{cases} \tag{4.4.1}$$

Here, $L_{existing} = 0.8$ and $k_{existing} = 0.1$ represent the limit and the growth rate of the logistic function, respectively. The variable $n = 1$ shifts the curve along the x-axis, influencing the distribution of probabilities across object counts.

Figure 24 illustrates the evolving probability of an object match across different object categories within the simulation. Initially, the probability of a match is high for all object categories, attributed to the broad and unrefined descriptive attributes employed at the simulation's

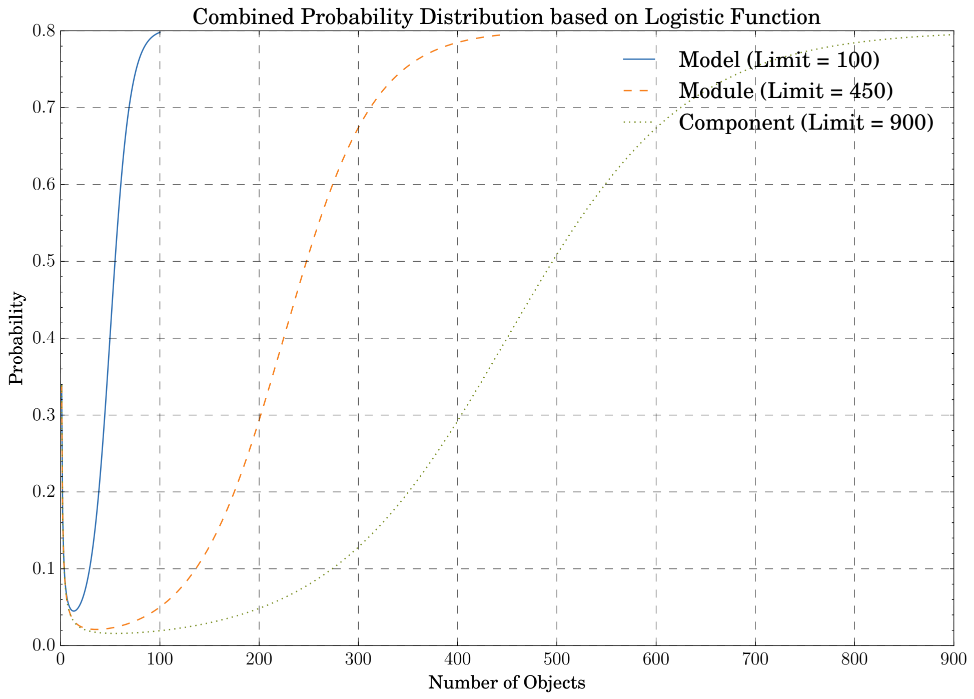


Figure 24: Combined probability distribution based on logistic function, with probability of object match per number of objects across three object categories: Model (blue), module (orange), component (green) (own figure).

commencement. This initial condition reflects a less discriminative matching process, where the specificity of attributes is minimal, thereby increasing the likelihood of matches. As the simulation progresses and the level of attribute refinement increases, the probability of finding a match decreases. This reduction is due to the increasing specificity and differentiation of attributes, which narrows the potential for matches. However, an interesting dynamic emerges as the simulation further evolves: the probability of a match begins to increase again with the growing number of objects. This latter trend is indicative of a more populated object space, where the sheer volume of objects enhances the likelihood of finding matches, even amidst more refined and specific attributes. This curve, depicted for each object category, underscores the complex interplay between attribute refinement and object proliferation within the simulation, revealing how these factors collectively influence the probability of object matches in a non-linear fashion.

- **Probability of Object Applicability**

The logistic function also calculates the probability of object applicability, as shown in Equation 4.4.2.

$$P_{applicable}(x) = \begin{cases} 0 & \text{if } x \leq 1, \\ \dfrac{L_{applicable}}{1+e^{-k_{applicable}\left(\frac{x-x_0}{\frac{data\_scale}{100}}\right)}} & \text{otherwise,} \end{cases} \tag{4.4.2}$$

This equation adjusts the probability of applicability based on the object's sequence or total count, with parameters tailored to reflect the complexity and prevalence of different object types within the simulation.

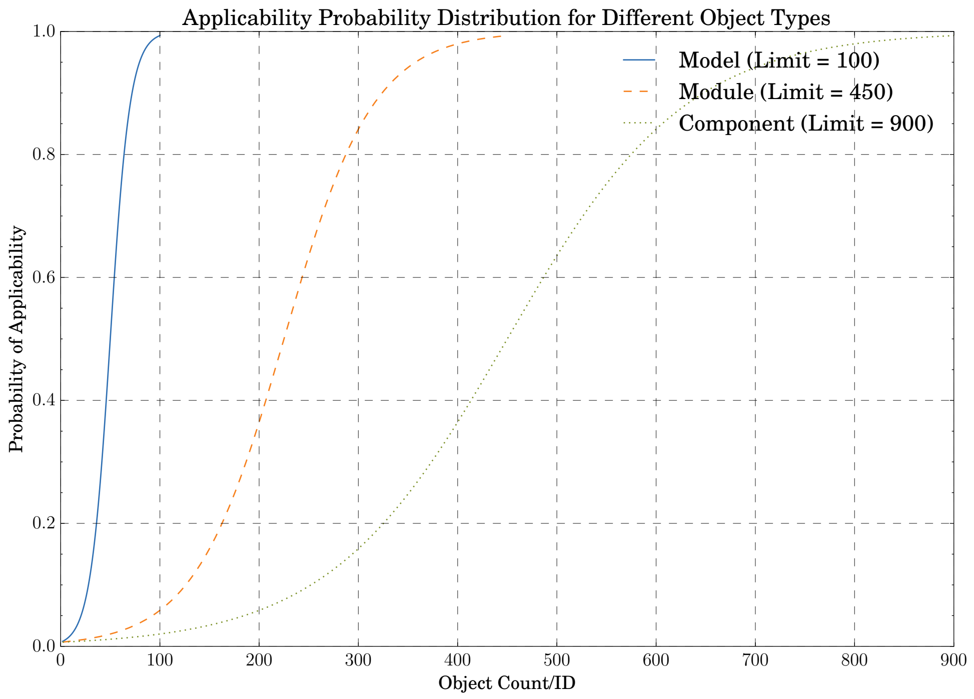


Figure 25: Applicability probability distribution for different object types, with probability of applicability per number of preexisting objects across three object categories: Model (blue), module (orange), component (green) (own figure).

Figure 25 illustrates the relationship between the cumulative number of preexisting objects within the simulation environment and the applicability probability of three distinct object categories: models, modules, and components. This visualization provides a clear depiction of how the probability of an object being applicable varies as a function of the number of objects already introduced into the simulation. Each curve in the figure corresponds to one of the object categories, illustrating that the probability of applicability is not uniform across categories but instead depends significantly on the object type and the existing object count. For models, the curve suggests a nuanced behavior where their applicability may be influenced by both their inherent complexity and the simulation's evolving state. Modules and components, meanwhile, display distinct patterns of applicability probability, reflecting their roles and integration within the simulation's architecture. The figure effectively demonstrates that the object applicability within the simulation is a dynamic attribute, influenced by the interplay between object type and the density of the simulation's object population, providing essential insights into the simulation model's behavior and its capacity to mimic complex systems dynamics.

- **Partial Object Applicability**

Adjusting for complexity factors, the probability of partial applicability is governed by Equation 4.4.3, which modifies the logistic function's steepness and distribution.

$$P_{applicable}(x; P\_S) = \frac{L}{n + \exp\left(-\frac{k}{\frac{data\_scale}{100}} \cdot \left(\frac{x}{P\_S} - x_0\right)\right)}, \tag{4.4.3}$$

This formula dynamically adjusts applicability probabilities, illustrating how complexity influences the simulation model's object interactions.

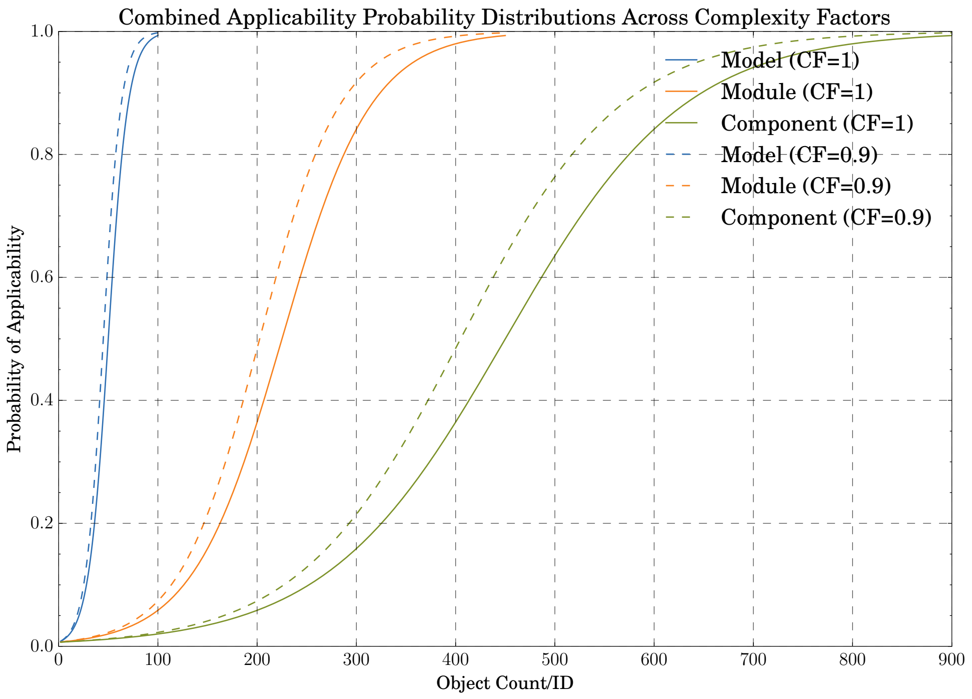


Figure 26: Combined partial applicability probability distributions across `PARTIAL_SCALE` parameter ranges, with probability of partial applicability per number of preexisting objects across three object categories: Model (blue), module (orange), component (green). Solid line (—): `PARTIAL_SCALE` = 1; Dashed Line (- -): `PARTIAL_SCALE` = 0.9 (own figure).

Figure 26 demonstrates the linear relationship between the `PARTIAL_SCALE` (P_S) static parameter and the applicability probability for each object type within the simulation. The figure illustrates this relationship through two distinct linestyles: a solid line (—) represents the initial applicability function, while a dashed line (- -) depicts the adjusted applicability values when the `PARTIAL_SCALE` parameter is set to 0.9. This visual distinction allows for an immediate comprehension of how adjustments to the `PARTIAL_SCALE` parameter linearly affect the probability of object applicability across all object types. The linear relationship suggests that as the `PARTIAL_SCALE` parameter decreases, there is a proportional reduction in the applicability probability for models, modules, and components, underscoring the parameter's vital role in modulating the simulation's dynamics.

- **Required Objects for Branch**

The number of required objects for a branch, especially for models, is determined by a stochastic process, detailed in Equation 4.4.4.

$$N_{required} = \begin{cases} \text{randint}(1, C) & \text{if object.type = “Model”}, \\ 1 & \text{otherwise}, \end{cases} \tag{4.4.4}$$

Here, $N_{required}$ depends on the model's complexity C, introducing variability into the branch structure of the simulation.

- **Model Complexity Generation**

The complexity of a model is generated through a random process defined by Equation 4.4.5.

$$C = \text{randint}(5, 10), \tag{4.4.5}$$

This approach allows for the simulation of models with varying degrees of complexity, enhancing the realism of the system dynamics.

These dynamic parameters and their mathematical formulations underpin the simulation model's capability to mimic complex real-world systems with a high degree of fidelity. Through stochastic processes and logistic functions, the model simulates the variability and complexity inherent in such systems, offering valuable insights into their behavior and interactions.

Model Workflow

The simulation model is comprised of three core object classes: `Model`, `Module`, and `Component`. These classes form a hierarchical structure that mimics the complexity and interdependencies found in real-world systems. Each class is designed with specific attributes and methods that allow for the dynamic evolution of the simulation over time, including the generation of complexity, interaction between objects, and the refinement of object attributes.

The `Model` class displayed in Listing (Lst.) 4.1 encapsulates the foundational aspects of model management within the framework, serving as a cornerstone for simulating complex systems. Each instance of `Model` is uniquely identified (`id`), allowing the traceability and individual assessment of models within the simulation environment. This class is instrumental in initializing models with an inherent complexity level, a vital attribute that directly impacts the model's simulation behavior and efficiency. Complexity governs the computational resources and time estimations associated with model operations, including validation checks, descriptive analyses, refinement processes, and branching strategies.

Furthermore, the `Model` class is designed to oversee the lifecycle of modules, embodying the modular design principle central to this work. By managing modules—each representing a discrete functional unit within a model—this class enhances the system's adaptability and scalability. The global tracking of created models, modules, and components ensures a cohesive operation across the distributed development environment, aligning with the framework's objective to rationalize PdM model development through modularization and efficient resource allocation.

The models' adaptability is further exemplified by the dynamic adjustment of their complexity. This feature allows for a responsive adjustment of time-related attributes (`time_to_check`, `time_to_describe`, `time_to_refine`, and `time_to_find_branch`), reflecting the model's operational efficiency and fitness within the simulation. The modular architecture, underscored by the management and composition of modules, provides a granular approach to model development and testing, essential for achieving high degrees of precision and reliability in simulation outcomes.

Lst. 4.1: Pseudocode for Model Class

```
Class Model
  Initialize:
    Increment global created_models counter
    Assign unique id based on created_models
    Set type to "Model"
    Initialize counters for various operations (check, refine, branch) for model,
  ↪ module, component levels
    Initialize list of modules
    Determine model complexity and update related timings

  Property complexity:
    Getter: Return model complexity
    Setter: Update model complexity and adjust related timings accordingly

  Method update_times:
    Update time-related attributes based on model complexity

  Method create_model:
    Create a new instance of Model
    Append to global models list
    Return new model instance

  Method remove_model(model):
    Remove a specified model from the global models list

  Method create_module(id = None):
    Create a new module with a specified id or based on the current number of modules
```

```
      Ensure uniqueness and append to the model's module list
      Return new module instance

    Method remove_module(module):
      Remove a specified module from the model's module list

```

Modules constitute a fundamental layer within the simulation framework, orchestrating the hierarchical structure that spans from models to the more granular components. Each instance of a `Module`, as detailed in Lst. 4.2, is inextricably linked to a parent model, thereby embedding itself within the larger context of the simulation's architecture. This hierarchical association is not merely structural but functional, allowing modules to serve as conduits through which components are created, managed, and aligned with the overarching objectives of the simulation. The attribute of complexity within modules mirrors the dynamic nature of models, where complexity not only defines the inherent attributes of a module but also influences its operational efficiency and interaction parameters with both parent models and nested components. This complexity is a mutable property, subject to adjustments as the simulation evolves, thereby offering a mechanism to finely tune the simulation's granularity and responsiveness.

Through the lens of the `Module` class, the simulation unveils a multi-layered approach to managing complexity. Modules are not static entities; they are active participants in the simulation's lifecycle, engaging in a continuous process of refinement and adaptation. Their ability to create and manage components further emphasizes their role as central elements in the simulation's architecture, enabling a modular and scalable approach to building sophisticated simulation environments. The actions of creating and removing components, as encapsulated in the methods of the `Module` class, highlight the dynamic interplay between structure and functionality within the simulation framework.

Lst. 4.2: Pseudocode for Module Class

```
  Class Module
    Initialize(parent_object, id):
      Increment global created_modules counter
      Construct module id from parent_object.id and given id
      Set type to "Module"
      Link to parent_object
      Initialize refinement counter and components list
      Set base complexity
      Call update_times to set initial time attributes

    Property complexity:
      Getter: Return the complexity
      Setter: Update the complexity and adjust time attributes accordingly

    Method update_times:
      Update time attributes (check, describe, refine, find branch) based on complexity
      Adjust property of fit inversely with complexity

    Method create_component(id = None):
      If id is not provided, use the number of existing components to assign a new id
      Else, extract and use the specified id
      Create a new Component instance with assigned id, setting its complexity to match
    ↪  the module's
      Add the new component to the module's components list if it is unique
      Return the new component instance

    Method remove_component(component):
      Remove the specified component from the module's components list

```

At the heart of the simulation's hierarchical structure lie the components, the most granular entities within the framework, as outlined in Lst. 4.3. Encapsulated within modules, components embody the nuanced facets of the simulation environment, thereby enriching the simulation's fidelity and enabling a precise modeling of elaborate system behaviors. The design of the `Component` class underscores this intent, enabling a detailed representation of environmental variables and interactions at the micro level.

Inheriting complexity from their parent module, components serve as central elements in the simulation, mirroring the complexity of real-world phenomena through their attributes and behaviors. This inheritance mechanism ensures that the component's operational dynamics are in alignment with the overarching module's characteristics, promoting a cohesive simulation experience. The adjustable complexity attribute of components, as presented in the pseudocode, not only influences their individual performance and interaction patterns but also allows for a dynamic adjustment of the simulation's granularity in response to evolving modeling requirements. Through their methodical management and the sophisticated interactions they support, components significantly contribute to the simulation's adaptability and depth. By offering a mechanism for fine-grained control over the simulation's dynamics, the `Component` class enhances the framework's ability to model complex systems with high precision. This granularity accelerates the exploration of specific hypotheses within the simulated environment, extending the utility and applicability of the simulation across diverse research and application domains.

Lst. 4.3: Pseudocode for Component Class

```
  Class Component
    Initialize(parent_object, id):
      Increment global created_components counter
      Construct component id from parent_object.id and given id
      Set type to "Component"
      Link to parent_object
      Initialize refinement counter
      Set base complexity
      Call update_times to set initial time attributes

    Property complexity:
      Getter: Return the complexity
      Setter: Update the complexity and adjust time attributes accordingly

    Method update_times:
```

```
    Update time attributes (check, describe, refine, find branch, create) based on
    ↪ complexity
    Adjust property of fit inversely with complexity

```

The hierarchical relationship between models, modules, and components allows for a detailed and nuanced simulation of complex systems. Each class contributes to the simulation's dynamic behavior, with complexity and interaction mechanisms designed to mimic real-world variability and interdependencies.

The `layer_workflow` function encapsulates a critical aspect of the simulation's operational logic, orchestrating the interactions and evolutionary processes among models, modules, and components. By dynamically evaluating the applicability of existing objects to the simulation's ongoing scenarios and objectives, this function plays a vital role in maintaining and adapting the simulation environment to align with specific requirements and contexts.

Lst. 4.4: Pseudocode for Layer Workflow Function

```
Function layer_workflow(env, object, models, existing_objects, applicable_objects,
↪ partial_applicable_objects, list_of_compared_objects = []):

  Initialize applicable_object as None
  Filter existing, applicable, and partial applicable objects by removing compared
  ↪ ones

  If there are existing objects:
    For each existing_object in existing_objects:
      Wait for existing_object's time_to_check
      Log check action

      Increment check counter for the object

      If existing_object is fully applicable:
        Set applicable_object to this object
        Log applicability and process termination

        If object is a Module:
          Append applicable_object to parent object's modules list
          Remove current object from parent object's modules list

        If object is a Component:
          Append applicable_object to parent object's components list
          Remove current object from parent object's components list

        Break from the loop

      Else if existing_object is partially applicable:
        Log partial applicability
        Wait for time to find branch

        Increment branch counter for the object
        Log branch identification

        Determine split_object based on object type
        Adjust complexity of split_object and object
        Wait for time to describe both objects
        Log description of both objects

        If object is not a Model:
          Create split object (module or component) with adjusted complexity

        Break from the loop

      Else:
        Log non-applicability
        Wait for time to refine the existing_object
        Increment refine counter for the existing_object
        Log refinement

    If no objects are partially applicable:
      Log no applicability
      Wait for time to refine the object
      Increment refine counter for the object
      Log refinement

  Else:
    Log the need for a new object type
    Reset list_of_compared_objects

```

Central to the `layer_workflow` function's operations are the comprehensive steps it undertakes to assess and modify the simulation's object landscape, as outlined in the associated pseudocode Lst. 4.4:

1. **Selective Evaluation:** The function commences by filtering previously evaluated objects from the pool of existing, applicable, and partially applicable objects. This ensures a focused and efficient assessment process, avoiding redundant evaluations and optimizing the workflow's computational resources.
2. **Determining Full Applicability:** It then proceeds to identify objects that fully satisfy the simulation's current needs. Fully applicable objects are consistently integrated into the simulation's hierarchical structure or replace less suitable objects, depending on their designated role (e.g., Module or Component). This integration marks a critical juncture in the workflow, potentially concluding the evaluation process for the current object if an optimal match is found.
3. **Addressing Partial Applicability:** In the absence of fully applicable objects, the function shifts its focus to partially applicable objects. These objects undergo a detailed branching and complexity adjustment process, tailored to refine their attributes and enhance their suitability for the simulation's requirements. This step embodies the simulation's adaptability, allowing for the creation of specialized object variants through a deliberate adjustment of their complexity levels.
4. **Refining Non-Applicable Objects:** Objects deemed neither fully nor partially applicable are subjected to a refinement process. This phase is aimed at incrementally enhancing their potential for future applicability, reflecting the simulation's iterative approach to optimizing its object portfolio for evolving simulation scenarios.
5. **Facilitating New Object Creation:** Finally, if the existing objects fail to meet the

required applicability criteria, the function signals the need for the creation of new objects. This aspect underscores the simulation's dynamic capacity to expand and evolve its object repository in response to new challenges and objectives, ensuring a resilient and responsive simulation environment.

The `layer_workflow` function thereby underlines the simulation's commitment to a dynamic, adaptive, and highly responsive modeling environment. Through this function, the simulation not only assesses and optimizes object applicability and functionality but also embraces the complexities and unpredictabilities inherent in modeling real-world phenomena. This adaptive mechanism ensures the continual evolution of the simulation, enabling it to address complex systems with nuanced and high-fidelity modeling capabilities.

Results

In conclusion, the simulation outcomes presented in Figure 27 demonstrate the runtime efficiency of the proposed model under default parameters across 50 runs, each encompassing the creation of 100 models.

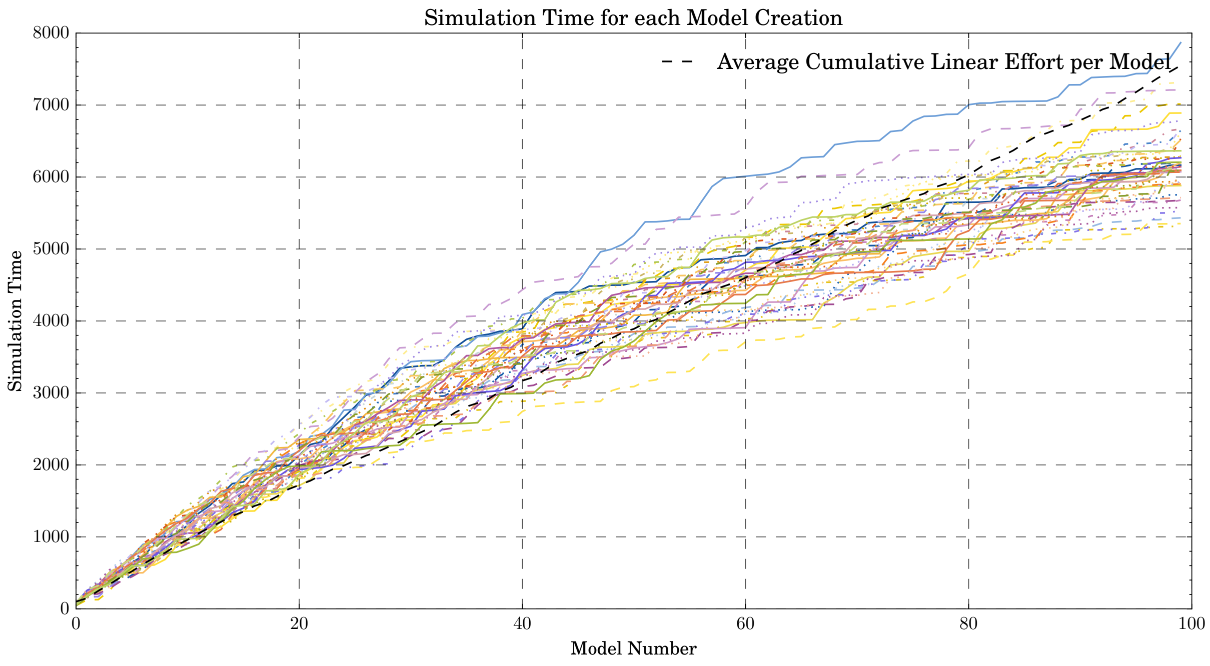


Figure 27: Runtime of 50 simulation runs with 100 models each, showing the simulation time per model number. Dashed black line: Average cumulative linear effort per model (own figure).

The cumulative runtime for each model within every simulation run is depicted, with the baseline linear model development approach, characterized by random model complexities, represented as a dashed black line. Initially, the simulation reveals that runtime for the early models generally exceeds that of the baseline. This trend persists until the creation of the 40th model, where the majority of simulation runs exhibit longer runtimes compared to the baseline. However, a notable shift occurs between the 40th and 50th models, during which the duration of the simulation runs begins to align more closely with the benchmark, subsequently showing a rapid improvement. Remarkably, from the 80th model onward, every simulation run achieves a shorter runtime than the benchmark, illustrating the efficiency and adaptability of the proposed model in optimizing the development process over successive iterations. This pattern underscores the potential of the proposed approach to not only match but significantly surpass the performance of traditional linear model development strategies, especially as the simulation progresses and the model refines its parameters and strategies.

A deeper examination of the simulation model's behavior is aided through the analysis presented in Figure 28, where individual model parameters are examined across all layers of the simulation model—Model, Module, and Component.

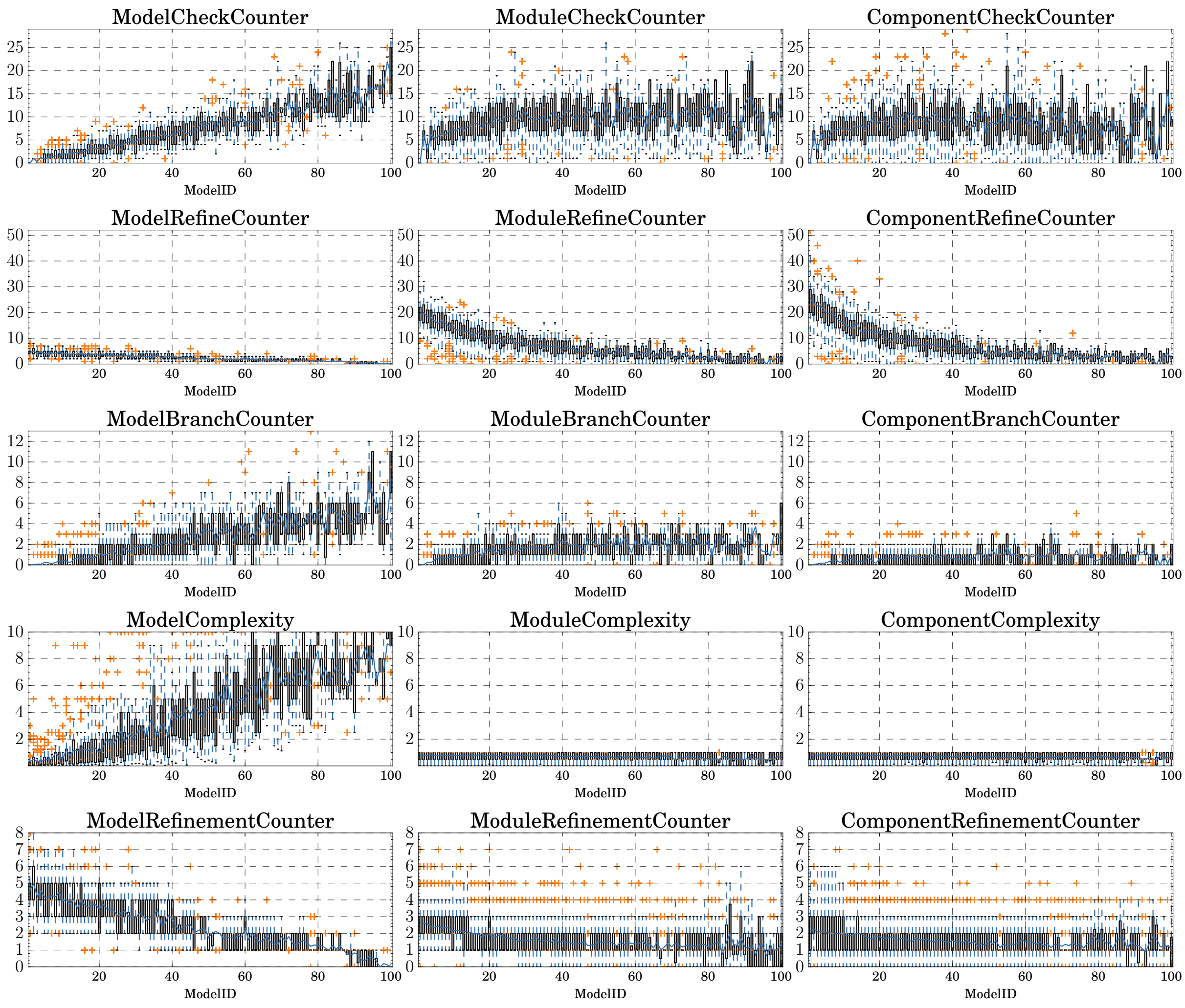


Figure 28: Distribution of model parameters across the model, module and component layers of the simulation model, showing aggregated boxplots for each Model ID over every run. Standardized axes for each parameter across the layers, mean values for each parameter depicted as blue line plots within every subplot (own figure).

This analysis employs aggregated boxplots for each model ID over every run, providing a comprehensive overview of the frequency of necessary checks, refinements, and branching actions, in addition to quantifying the individual object complexities and degrees of refinement. Importantly, the axes for each parameter are standardized across the layers, enabling a direct comparison of behaviors and trends within the Model, Module, and Component layers. Additionally, the mean values for each parameter are depicted as blue line plots within every subplot, significantly enhancing the visualization of underlying trends and providing a clear reference point for assessing average behaviors across simulations. This standardized approach reveals nuanced insights into the operational dynamics of the simulation model, highlighting the interplay between the complexity of objects and the efficiency of refinement and adaptation processes. Through this detailed examination, patterns emerge that underscore the adaptive mechanisms at play within the simulation, illustrating how different layers respond to the demands of simulation and evolve over time. The fixed axes for every layer ensure that observations are not only consistent but also comparable, providing a clear picture of how checks, refinements, and branches contribute to the overall behavior and efficiency of the model across its various components and layers.

The **check** action counter serves as a critical metric within the simulation model, quantifying the necessity for developers to engage in verification processes to ascertain the actual applicability of objects against specified requirements. The analysis of model checking processes reveals a near-linear distribution, escalating from an initial state of zero checks to an average of 15 checks for models. This progression signifies that, within a repository containing approximately 100 models, about 15 models require examination to match the given description before determining the feasibility of reuse or the necessity to innovate a new model. For modules, the distribution of checks exhibits a growth from zero to an approximate average of 10 checks, displaying an almost asymptotic trend over the creation of the first 30 models. Subsequently, this trend stabilizes, indicating a plateau in the necessity for additional checks beyond this point. Intriguingly, the component check frequency initiates with a similar asymptotic increase towards 10 checks but then demonstrates a gradual decline as the number of models approaches 100. This nuanced behavior suggests a diminishing requirement for component-level verification in later stages of the simulation, possibly reflecting an increased efficiency in component applicability or a saturation in the diversity of components that call for evaluation.

The analysis of the **refinement** counter unveils distinct patterns in the iterative refinement process across the various layers of the simulation model. For the model layer, the refinement activity exhibits an almost linear trend, commencing with an average of 5 refinement iterations for the initial models. This trend suggests a pronounced need for refinement in the early stages of model development, which gradually diminishes to between zero and one refinement step for the concluding models. This reduction implies a maturation in the precision of model descriptions over time, minimizing the necessity for further refinement. Conversely, both the Module and Component layers demonstrate a markedly different pattern, characterized by an exponential decrease in refinement activities. The number of refinement processes for modules and components asymptotically approaches 1 to 2 for the final models, indicating a significant reduction in the need for iterative refinement. The starting point for module refinement is approximately 20 processes, while component refinement begins at around 25 processes. It is to be emphasized that these figures represent the total number of refinement actions per model-related object within the model creation process and do not directly indicate the degree of refinement, which will be addressed separately. This differential in refinement patterns across layers highlights the adaptive efficiency of the simulation model, with initial intensive refinement giving way to a more structured process as the simulation progresses and the system's descriptions become increasingly precise.

The **branch** counter serves as an essential metric for understanding the dynamics of branching processes within the simulation model, where branching signifies the identification of partial applicability and the subsequent determination of a branch point by developers. In the context of Model branching, an interesting trend is observed, with the counter indicating an incremental growth in the number of branched models in correlation with the total number of models. This trend suggests an increasing propensity for models to undergo branching as the simulation progresses, reinforcing the premise that models within the simulation are primarily evaluated on the basis of partial applicability or direct match to the requirements descriptions. While this approach simplifies the branching narrative, such simplification does not detract from the overall simulation outcomes, as illustrated in the workflow model discussion in subsection 4.4. Intriguingly, contrary to the trend observed in the model layer, the branching processes for both modules and components exhibit a near-constant rate, suggesting a stabilization in the frequency of partial applicability scenarios for these layers. This observation underscores a divergent behavior between the model layer and the subordinate module and component layers, with the latter demonstrating a steadier pattern of branching activity. Such a disparity highlights the nuanced complexities inherent in the simulation's operational mechanics, where the propensity for branching reflects the varying levels of adaptability and applicability challenges encountered across different simulation layers.

The concept of **object complexity** lies at the heart of the simulation's dynamic refinement process, with branching serving as a central mechanism for reducing complexity by dividing a single object into two entities of lower complexity. This split permits more targeted and precise refinement efforts. Notably, the complexity of models exhibits a trend reminiscent of a symmetric logistic curve, indicating a significant reduction in complexity for early models, which are frequently refined and branched. Initially started from a baseline complexity range between 5 and 10, the early models quickly converge to a very low complexity level of less than 1. This rapid complexity reduction underscores the effectiveness of branching and refinement in honing the model's applicability and precision.

In contrast, both Modules and Components are characterized by a lack of variation in their assumed initial complexity, starting uniformly at 1. A phenomenon of particular interest observed in these layers is the negative development of average complexity over time. This trend suggests a prevailing tendency within the simulation to branch objects that have undergone extensive refinement and detail enhancement, thereby prioritizing the division of highly specialized and elaborately developed objects. Such a pattern reflects a strategic emphasis on the branching of objects that, due to their refined state, offer a more nuanced and detailed basis for generating two distinct entities with inherently lower complexities.

The analysis of object complexity and its evolution through branching and refinement processes highlights the nuanced strategies employed within the simulation to optimize object applicability and functional precision. The observed trends across different layers—especially the distinct behavior between models and the module/component layers—illustrate the layered approach to complexity management and the simulation's overarching goals of efficiency and adaptability in model development.

The **refinement score**, indicative of the average number of refinement iterations each object undergoes in the model, illuminates significant aspects of the simulation's refinement dynamics. A notable observation across all object types—Models, Modules, and Components—is the marked decrease in the average number of refinements required over time. This declining trend encapsulates two vital factors: the cumulative effect of object checks and the progressive enhancement in the level of object refinement.

For Modules and Components, the behavior is strikingly similar, manifesting a slight asymptotic trend. This pattern suggests a gradual approach to an equilibrium state where the necessity for further refinement stabilizes, reflecting the maturation of the simulation environment and the refinement process itself. These objects exhibit a nuanced balance between the need for refinement and the attainment of a refined state, highlighting the efficiency of the simulation's adaptive mechanisms.

Conversely, the model refinement counter showcases a more pronounced, almost linear decline. This trend signifies a rapid convergence towards minimal refinement needs for models, underscoring the precision of initial model descriptions and the effectiveness of early refinement efforts. The linear trajectory of this decline points to a straightforward reduction in refinement complexity, suggesting that models quickly reach a state of refinement that negates the need for extensive iterative refinement processes.

These observations collectively underscore the simulation model's capacity for self-improvement and adaptation. As the simulation progresses, the diminishing need for refinement across models, modules, and components not only signals an increasing efficiency in meeting simulation requirements but also reflects the cumulative learning and enhancement embedded within the simulation process. The nuanced differences in refinement trends between object types further highlight the layered complexity of the simulation model, with each layer evolving towards optimization in a manner reflective of its unique operational dynamics and challenges.

In summation, the analysis presented herein not only validates the simulation's accurate depiction of the envisioned workflow, affirming its utility in yielding substantive insights, but also unveils intriguing findings regarding the elaborate interplay of parameters within the simulation. These observations explicitly highlight the inherent complexity of parameter interactions, serving to accentuate the indispensable role of a sophisticated complexity-management methodology. The employment of event-discrete simulation emerges as a vital strategy in this context, providing a robust framework to navigate and define the complexities inherent in the simulation. Such a methodological approach is not merely beneficial but essential in ensuring the fidelity and applicability of the simulation model to real-world scenarios, thereby reinforcing the significance of adopting event-discrete simulation techniques in the development and refinement of complex simulation workflows. The results, therefore, not only validate the correct behavior of the simulation—thereby endorsing its applicability for generating profound insights into the constructed workflow—but also underscore the complexities of parameter interferences, thereby underlining the necessity for a methodical approach to complexity handling, such as that offered by event-discrete simulation.

4.4.3 Results of the Sensitivity Analysis Evaluation

Within the scope of this work, a comprehensive sensitivity analysis is conducted on the newly developed framework designed to support the development of PdM models by distributed teams. The objective of this sensitivity analysis is to thoroughly assess the framework's robustness and identify the critical factors that significantly influence its performance and effectiveness. Sensitivity analysis emerges as a central component in this process, offering a systematic approach to evaluate how variations in the framework's parameters affect the efficiency and outcome of the PdM model development process. This is particularly important in the context of distributed team environments, where the coordination of tasks and integration of components are subject to variability and uncertainties. The analysis aims to highlight the framework's adaptability to different operational scenarios and its resilience to potential disruptions, thereby ensuring that the framework can serve as a reliable foundation for faster and more collaborative development of PdM models. Through this sensitivity analysis, insights are gained into optimizing the framework's structure and workflow, ultimately enabling a more efficient and effective approach to PdM model development in distributed settings.

The Sobol method, a variance-based sensitivity analysis technique, is highly effective for investigating the influence of input variables on the output of complex models, such as event discrete simulation models in PdM AI (Sobol, 2001). This method employs a quasi-random sampling technique to generate model inputs and decomposes the variance of the output into components attributable to different inputs or sets of inputs.

The Sobol method stands out among various sensitivity analysis techniques due to its comprehensive variance-based approach, which contrasts sharply with local methods such as One-at-a-Time (OaT), where each parameter is varied individually while holding others at their baseline levels. Unlike OaT, which can overlook interactions between parameters, the Sobol method evaluates the impact of interactions through its higher-order indices, providing a more detailed understanding of system behavior (Saltelli, 2002). Another commonly used method, the Morris method, involves a more qualitative measure of sensitivity and is simpler but less detailed than the Sobol method. The Morris method screens for influential factors through a one-step-at-a-time strategy, which, although efficient, lacks the depth provided by the variance decomposition in the Sobol method. Additionally, while derivative-based methods such as the adjoint approach offer precise gradients that are useful for smooth and continuous systems, they struggle with discontinuities and non-linearities where the Sobol method excels. Therefore, the Sobol method is especially advantageous for complex PdM models where the outputs are highly nonlinear or non-monotonic in response to input variations, affirming its suitability in distributed industrial applications (Sobol, 2001; Saltelli, 2002).

The Sobol method is based on the decomposition of the model output Y as a function f of n input variables:

$$Y = f(X_1, X_2, \ldots, X_n) \tag{4.4.6}$$

This output can be decomposed into terms of increasing dimensionality:

$$V = \sum_{i=1}^{n} V_i + \sum_{i<j} V_{ij} + \cdots + V_{1,2,\ldots,n} \tag{4.4.7}$$

where V is the total variance of Y, V_i is the variance caused by the input X_i alone, and V_{ij} is the variance caused by the interaction between X_i and X_j, and so forth.

The Sobol indices, which are derived from this decomposition, are used to quantify the contribution of each input to the total variance. The first-order Sobol index for an input X_i is defined as:

$$S_i = \frac{V_i}{V} \tag{4.4.8}$$

This index represents the proportion of the variance of Y that is due to X_i alone. Higher-order indices can similarly be calculated to assess interactions between multiple inputs. The sum of all Sobol indices, including higher-order interactions, is equal to 1.

The strength of the Sobol method lies in its ability to handle non-linear and non-monotonic relationships between inputs and outputs, making it particularly suitable for complex PdM models employed in distributed industrial environments (Sobol, 2001).

The Sobol sensitivity analysis method, a prominent global sensitivity analysis technique, employs variance-based decompositions to quantify the contributions of each input parameter to the output variance of a model. The Sobol indices are derived from the variance decomposition of the model output Y, which can be expressed as:

$$Y = f(X_1, X_2, \ldots, X_n) \tag{4.4.9}$$

where f represents the model, and $X_1, X_2, \ldots, X_n$ are the input parameters. The total variance V of the output Y can be decomposed as:

$$V = \sum_{i=1}^{n} V_i + \sum_{i<j} V_{ij} + \ldots + V_{1,2,\ldots,n} \tag{4.4.10}$$

where V_i is the variance contribution from each individual parameter and V_{ij} represents the interaction effects.

Convergence of the Sobol indices with increasing sample size N is essential for ensuring their reliability. Stability of these indices, as N increases, suggests adequate sampling and convergence:

$$\lim_{N\to\infty} S_i(N) = S_i \tag{4.4.11}$$

where $S_i(N)$ represents the estimate of the Sobol index for parameter X_i at sample size N.

To assess the uncertainty in the Sobol indices, bootstrap resampling techniques are employed, allowing the calculation of confidence intervals for each index, thus providing a statistical measure of the indices' convergence (Saltelli, Annoni, et al., 2010).

Adaptive sampling algorithms are used to optimize resource allocation by targeting inputs with significant uncertainty in their sensitivity indices, which can accelerate the convergence process (Homma and Saltelli, 1996; Sobol, 2001).

The seminal works by Sobol, Saltelli, and Homma are vital in framing the methodologies for evaluating the convergence of Sobol sensitivity indices, offering in-depth insights and advanced strategies in the field of global sensitivity analysis. Saltelli, Annoni, et al. (2010) provided a comprehensive overview of variance-based sensitivity analysis, thoroughly outlining the design protocols and estimator formulations for computing the total sensitivity index, which are crucial for understanding how model output variability can be attributed to different input variables. Earlier, Sobol (2001) introduced the now widely-used Sobol indices, detailing their calculation methods and discussing the critical aspects of their convergence, which is essential for ensuring the robustness and reliability of sensitivity analyses in complex mathematical models. Furthermore, the collaborative work of Homma and Saltelli (1996) examines the importance measures within global sensitivity analysis, particularly focusing on nonlinear models, and highlights methods to assess the impact of each model input under varying scenarios. These foundational texts collectively enhance the analytical toolkit available for researchers engaging in sensitivity analysis, especially in scenarios where model outputs are highly sensitive to input variations.

In the sensitivity analysis conducted on this work's artifact, a focused approach is adopted by analyzing only a subset of parameters from the model layer along with the global parameters, owing to the homogeneous nature of the object layers within the system. This methodological decision is underpinned by the assumption that the similar behavior and characteristics of these object layers imply that insights gained from a selected set of parameters can be extrapolated to the entire model. This strategy not only rationalizes the analysis by reducing computational complexity but also ensures that the results remain broadly applicable across the full scope of the model. Such a targeted approach in sensitivity analysis is supported by literature that emphasizes the efficiency and effectiveness of analyzing representative subsets in complex systems, where redundant or parallel structures exist (Saltelli, Annoni, et al., 2010; Sobol, 2001). This practice aligns with established methodologies which suggest that focusing on critical parameters can provide significant insights into the system's behavior without the necessity to examine every variable individually (Saltelli, Ratto, et al., 2007; Iooss and Lemaître, 2015).

In Figure 29, the results of the sensitivity analysis are depicted, showcasing the variation in Sobol indices 'ST' and 'S1' across a range of chosen sample sizes N set to [2, 4, 8, 16, 32, 64]. The selection of these specific sample sizes, each being a power of two, is primarily driven by computational considerations inherent to the quasi-random sampling methods used in the Sobol sensitivity analysis. Utilizing sample sizes that are powers of two enhances the efficiency of variance reduction techniques, such as the Sobol sequence, which require dyadic partitions of the input space to optimize the uniformity and distribution of sample points. This methodological choice ensures that the estimation of sensitivity indices is not only more efficient but also statistically robust, promoting a clearer and more accurate convergence assessment of the indices as the sample size increases. These sizes enable the Sobol method to systematically reduce error and variance in the estimates, which is central for achieving reliable and stable sensitivity analysis results (Sobol, 2001; Saltelli, 2002).

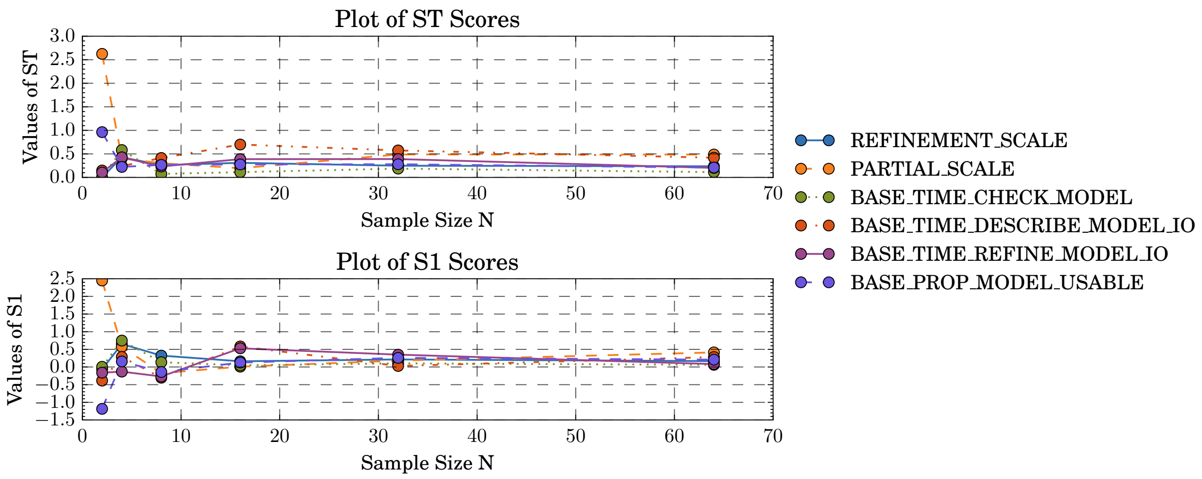


Figure 29: Plot of ST and S1 scores for parameters per N. Blue parameter: `Refinement_Scale`; Orange parameter: `Partial_Scale`; Green parameter: `Base_Time_Check_Model`; Red parameter: `Base_Time_Describe_Model_IO`; Aubergine parameter: `Base_Time_Refine_Model_IO`; Indigo parameter: `Base_Prop_Model_Usable` (own figure).

Convergence in the sensitivity analysis is evident from the stabilization of the Sobol indices across increasing sample sizes, as illustrated in Figure 29. Notably, the parameters `PARTIAL SCALE` and `BASE TIME DESCRIBE MODEL IO` emerge as having the most substantial impact on the overall results. This observation underscores the critical influence of these parameters on the model's output variability. The significant impact of `PARTIAL SCALE` suggests that even slight variations in this parameter can lead to considerable changes in the model outcomes, indicating a high sensitivity. Similarly, the `BASE TIME DESCRIBE MODEL IO` parameter also shows pronounced sensitivity, affecting the model's performance and accuracy. The clear visualization of these trends in the plot confirms not only the convergence of the sensitivity indices but also highlights the areas where model adjustments could yield the most substantial improvements in model reliability and predictiveness (Saltelli, 2002).

The findings from the sensitivity analysis, particularly the significant influence of the parameters `PARTIAL SCALE` and `BASE TIME DESCRIBE MODEL IO`, underscore the crucial role of the Formal Attribute Description System in the overall model framework, as this system directly impacts these parameters. This emphasis is not merely coincidental but indicative of the foundational importance of the attribute description system within the model's architecture. Detailed in Section 4.5, the Formal Attribute Description System facilitates a structured and precise definition of model attributes, which in turn significantly affects the model's sensitivity to input variations. The impact of this system is thus vital, as it substantiates the model's behavior and its analytical robustness. The clear correlation between these key parameters and the model's output variability highlights the effectiveness and critical nature of the attribute system in shaping the model's outcomes and its reliability. The enhanced understanding of these dynamics, as presented in the referenced section, offers vital insights into optimizing the model for better performance and more accurate predictions.

4.5 Descriptive Attributes System for Components

This work introduces a formal attribute system that addresses the shortcomings in existing research on the formalization of pipeline objects in PdM models. Particularly, the results from the sensitivity analysis in section 4.4.2 highlight the critical need for a structured approach to compare and evaluate the functionality of these objects. Despite the evident importance, research to date inadequately addresses the formalization of such systems within industrial applications. This subsection describes the development of a comprehensive attribute system that not only enhances the comparability and functional assessment of pipeline objects but also aligns with the agile and modular nature of the proposed framework. The formal attribute system is designed to consistently integrate into the existing PdM development processes, providing a robust methodological foundation that fosters precision and reliability in handling pipeline objects.

The *Z* Specification language, developed in the 1980s, is a foundational tool in formally specifying and verifying software systems, emphasizing the need for stringent methodologies in system development (Spivey, 1989; Woodcock and Davies, 1996). Similarly, advancements in automated code summarization show how formalized description systems can significantly enhance the comprehensibility and maintainability of complex codebases through the use of AI and ML techniques (Allamanis et al., 2017; LeClair, Haque, et al., 2020). Furthermore, recent developments in data description systems in data science advance robust methodologies for data preparation and exploration, highlighting the need for dynamic systems that adapt to new data analytics paradigms (Paganelli et al., 2020; Bortolotti, 2018).

The formal attribute system proposed in this subsection aims to integrate these principles, providing a robust methodological foundation that not only enhances the reliability and security of PdM models but also supports the dynamic nature of industrial applications. By leveraging the stringent specifications capabilities of *Z* and incorporating the adaptive qualities of modern code summarization and data description systems, this system will enhance both the precision and adaptability of pipeline object management in PdM.

The *Z* Specification language, developed in the 1980s, is a formal language used for describing and modeling computing systems. It is particularly noted for its use of mathematical notation to specify the behavior of systems in a precise and unambiguous manner. The foundation of *Z* lies in set theory and first-order predicate logic, which allows the stringent specification of system properties, including data structures and operations. One of the key features of *Z* is its support for the specification of abstract data types, enabling the formal definition of system states and the operations that can be performed on them. This capability is essential for the verification and validation of complex software systems, as it allows for the detailed analysis of system behavior before implementation. The use of *Z* in the software development process can significantly enhance the reliability and robustness of the resulting systems by enabling the early detection and correction of design errors. The literature on *Z* includes comprehensive texts such as *The Z Notation: A Reference Manual* by Spivey (1989), which provides an in-depth exploration of the language's syntax, semantics, and usage in system specification. Additionally, *Using Z: Specification, Refinement, and Proof* by Woodcock and Davies (1996) further elaborates on the practical aspects of applying *Z* in the context of software engineering, including refinement techniques and proof strategies to ensure that specifications accurately reflect the intended system behavior.

Despite its origins in the 1980s, the *Z* Specification language continues to be relevant in contemporary software engineering, particularly in the domains requiring high assurance and formal verification. Recent research focuses on integrating *Z* with modern software development practices, including agile methodologies and model-driven engineering, to leverage its stringent specification capabilities while maintaining flexibility and efficiency in the development process. For instance, advancements in tool support, such as the Z/Eves and CZT (Community *Z* Tools), enable the use of *Z* in verifying the correctness of software designs and automatically generating test cases, thereby enhancing the language's applicability to current software engineering challenges (Utting and Legeard, 2007; Zander, Schieferdecker, and Mosterman, 2017).

Moreover, the *Z* language is applied in critical systems where safety, security, and reliability are highly important. Examples include its use in the specification and verification of railway signaling systems, air traffic control software, and cryptographic protocols. These applications demonstrate the language's enduring value in projects where the cost of failure is high, and formal methods can provide significant benefits in terms of risk mitigation and assurance of system properties (Woodcock, Larsen, et al., 2009; Freitas and McDermott, 2011).

The ongoing development of *Z* and its ecosystem reflects a broader trend in the software industry towards the adoption of formal methods as a means to achieve higher levels of system reliability and security, especially in the context of increasing system complexity and the growing prevalence of CPSs.

Automated code summarization is an emerging field in software engineering and AI that addresses the growing need for efficient interpretation and documentation of increasingly complex codebases. As software systems become more sophisticated, understanding and maintaining code becomes a significant challenge. Automated summarization tools aim to alleviate this burden by providing concise, human-readable summaries of code functionalities, thereby enhancing code comprehensibility, maintainability, and developer productivity (Allamanis et al., 2017).

Recent advancements in this field have leveraged sophisticated AI techniques, particularly ML and NLP, to automatically generate summaries of code segments. A notable approach involves the use of Graph Neural Networks (GNNs), which are well-suited to represent the hierarchical and interconnected structure of code in the form of Abstract Syntax Trees (ASTs) (LeClair, Jiang, and McMillan, 2019). The work on *Improved Code Summarization via a Graph Neural Network* by LeClair, Haque, et al. (2020) demonstrates the effectiveness of GNNs in capturing the structural nuances of code for better summarization (LeClair, Haque, et al., 2020).

The state of the art in automated code summarization includes the integration of advanced neural network architectures, such as Transformer models, with GNNs to enhance the understanding of both the syntactic and semantic aspects of code (P. Fernandes, Allamanis, and Brockschmidt, 2018). Recent works like *Transformer-XL With Graph Neural Network for Source Code Summarization* by X. Zhang et al. (2021) exemplify this trend, showcasing improved performance in code summarization tasks. Additionally, the field is evolving to include context-aware summarization techniques, where the context of the code, such as its usage in a larger codebase or its functionality, is considered to generate more relevant summaries (Xing Hu et al., 2018). Models like CoCoSum utilize multi-relational GNNs for contextual code summarization (Yanlin Wang et al., 2021).

Despite these advancements, a notable gap persists in the development of a formalized and standardized description system that evolves over time. Current research primarily focuses on static aspects of code summarization, lacking a dynamic framework that adapts to the evolving nature of code and programming practices. This gap underscores the need for a system that not only summarizes code effectively but also evolves with changing coding paradigms and technologies, ensuring continued relevance and accuracy in code summarization.

Recent advancements in data description systems significantly enhance the capabilities of data analysis and data science. Paganelli et al. (2020) introduce an approach for generating insightful data descriptions through boolean predicates, accelerating interactive data exploration and anomaly detection. Bortolotti (2018) emphasizes the importance of data preparation, including descriptive statistics and variable analysis, as foundational for effective data modeling. Castro et al. (2015) explore the use of ontologies for research data description, demonstrating their application in vehicle simulation to ensure early and accurate data description. Barlas, Lanning, and Heavey (2015) survey open-source data science tools, proposing a classification scheme that highlights the current state-of-the-art tools and their contributions to the field. Furthermore, Karpatne et al. (2017) discuss the emerging paradigm of theory-guided data science (TGDS), which integrates scientific knowledge into data science models to enhance scientific discovery and ensure model generalizability. These studies collectively underscore the evolving nature of data description systems and their central role in advancing data analysis and science practices, by providing robust methodologies for data preparation, exploration, and the integration of domain-specific knowledge.

The evolution of data analytics increasingly favors the adoption of pipeline architectures, reflecting a paradigm shift towards more structured and efficient data processing methodologies. This trend underscores the critical need for formal description systems capable of articulating the complex interdependencies and operational semantics inherent in these pipelines. While the architecture of pipelines and databases considerably progresses in formalization efforts, the specific domain of data analytics pipelines remains relatively underexplored. The inherent complexity of encapsulating the functionality of data analytics pipelines, recognized as an NP-Hard problem, emphasizes this challenge.

In their seminal 2017 article, *A Formal Semantics for Data Analytics Pipelines*, Drocco et al. (2017) introduce a novel programming model designed to enhance the development of data analytics applications. Centered around the concepts of Pipelines and Operators, this model serves as the foundation for PiCo, a Domain-Specific Language tailored for Data Analytics Pipelines. The authors propose a distinctive approach where pipelines are conceptualized as workflows processing data collections, diverging from the traditional computational process perspective. A key innovation of their model is the polymorphism of PiCo operators with respect to data types, enabling the reuse of algorithms and pipelines across various data models such as streams, lists, and sets. This flexibility simplifies the update process of pipelines and allows abstract reasoning about programs, marking a significant advancement in the field of data analytics pipeline design.

The absence of a heuristic approach for the demand-driven description of pipeline objects marks a significant gap in the literature. The work of Drocco et al. (2017) presents a pioneering effort towards addressing this gap, offering a foundational framework for the conceptualization and operationalization of pipelines in data analytics. However, the development of a heuristic, demand-driven description system that can dynamically adapt to the nuanced requirements of data analytics use cases remains an open area for research. Such a system would not only aid the design and optimization of analytics pipelines but also enhance their interpretability and reusability.

Moreover, the introduction of debugging tools like Dagger for data-centric errors in pipelines and the exploration of heterogeneous execution environments further illustrate the complexities involved in pipeline management and the need for sophisticated support systems (Rezig et al., 2020; D. Wu et al., 2016).

In summary, the enduring relevance of the *Z* Specification language, alongside recent advancements in automated code summarization and the evolving trends in data description systems, underscores the need for a sophisticated, formal attribute system. These fields collectively demonstrate the critical role that precise, dynamic, and formalized systems play in enhancing system reliability, comprehensibility, and adaptability. Notably, the application of the *Z* language in critical systems and its integration with modern software practices highlight the benefits of formal methods in complex environments. Likewise, the progression in automated code summarization, particularly through advanced AI techniques, points to the necessity of evolving description systems that adapt over time to new coding paradigms. Similarly, the advancements in data description systems for data science emphasize the importance of robust methodologies that cater to the dynamic nature of data analytics.

Building on these insights, this work addresses a significant research gap by developing a formal attribute system designed to compare and refine the functionality of pipeline objects in PdM models. This system leverages the principles of formal specification, drawing inspiration from the *Z* Specification language, and incorporates the adaptability seen in modern code summarization and data description practices. The proposed attribute system aims to provide a dynamic, stringent, and standardized approach to manage and evaluate pipeline objects, ensuring their functional integrity and operational efficiency in distributed and complex industrial applications.

4.5.1 Visualization of the Descriptive Attributes System

To underscore the practical applicability and usefulness of the descriptive attributes system within the *Predictive Maintenance Framework*, Figure 30 presents a real-world example use case. This illustration serves not only as a validation of the descriptive attributes system but also as a demonstration of the model architecture and workflow in action. Through an initial walkthrough of the application, the figure illustrates how the system enables the functionality of PdM models in industrial applications.

Subsequently, the figure explores various scenarios stemming from the initial use case. These scenarios are thoroughly chosen to showcase the distribution and reusability effects inherent in the PdM component repository. By presenting different scenarios, the figure vividly demonstrates the flexibility and efficiency gains achievable through the application of the descriptive attributes system in managing and deploying PdM models across diverse industrial contexts.

In essence, Figure 30 acts as a microcosm of the broader *Predictive Maintenance Framework*, illustrating how descriptive attributes contribute to the development, deployment, and scalability of PdM applications. It exemplifies the core principles of reusability and modularization, which are vital to reducing development costs and enhancing efficiency in the creation of PdM models for vehicle components and beyond.

The objective of this example use case is to illustrate the process of deriving feature importance for vehicle speed in a lap-based racing scenario. This scenario is particularly relevant to the *Predictive Maintenance Framework* due to its emphasis on optimizing performance and maintenance strategies in high-stress, competitive environments.

The data for this use case comprises onboard logging from multiple vehicle signals captured over the duration of approximately 3 hours, encompassing around 60 laps, with a measurement frequency of 200Hz. This results in a comprehensive dataset of 205 signals, representing a rich source of information for analysis. Additionally, metadata derived from expert knowledge is incorporated to enhance the understanding and interpretation of the raw data signals.

A regression model is employed to analyze the collected data, with the aim of training and validating a model capable of returning a feature scoring list. This list is fundamental for identifying which signals are most influential in determining vehicle speed during the race, thereby providing insights into performance optimization and maintenance needs.

The primary output of this analysis is a list of features, each accompanied by corresponding feature importance scores. These scores are essential for understanding the relative impact of each signal on vehicle performance. Furthermore, the confidence of the ML model, derived from thorough ME, is presented alongside the feature scores. This confidence metric offers an additional layer of insight, indicating the reliability of the model's predictions and the robustness of the underlying data analysis process.

This use case stands as a testament to the potential of PdM applications in high-performance settings, demonstrating how data-driven insights can inform and enhance maintenance strategies, ultimately leading to improved vehicle reliability and efficiency.

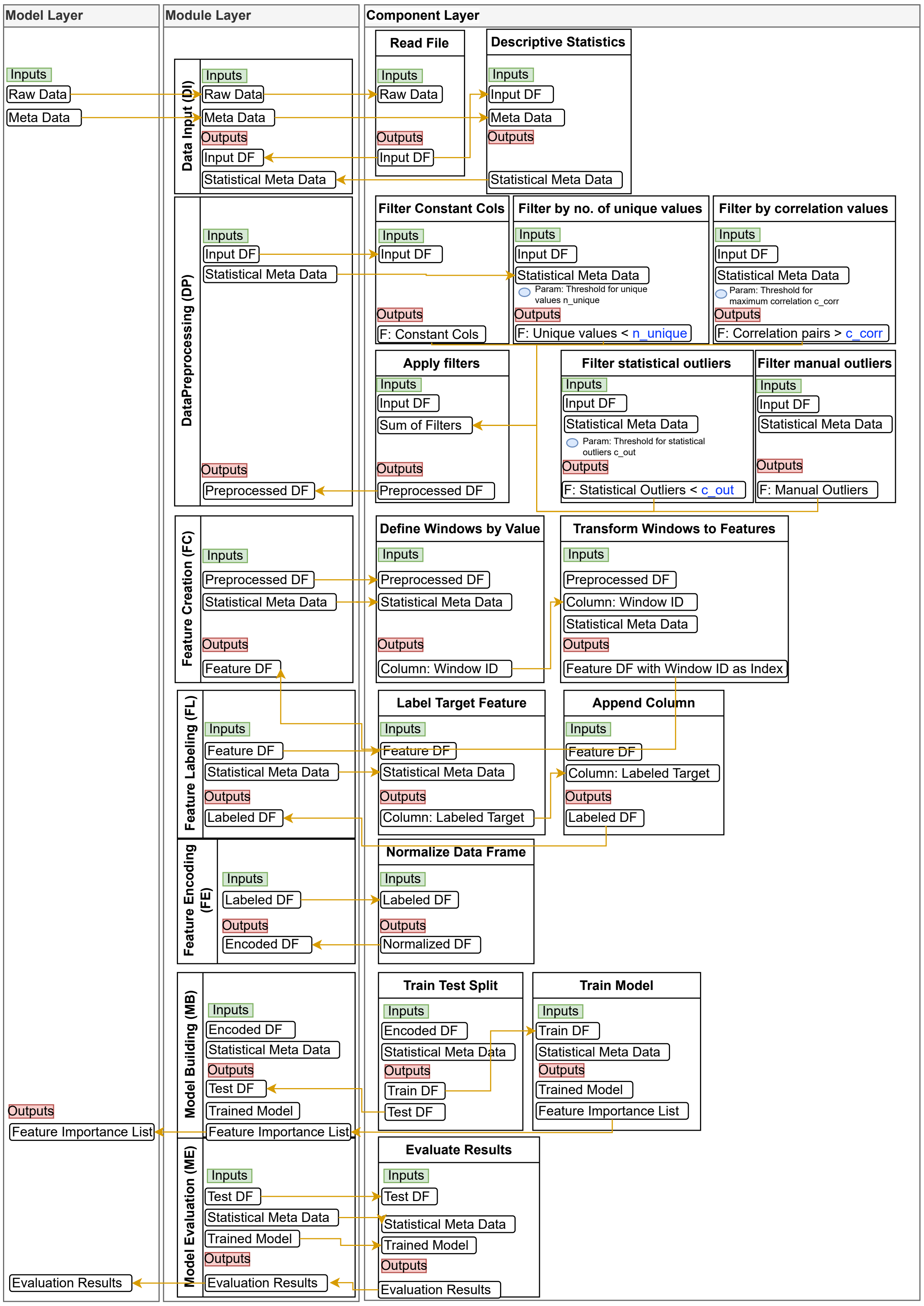


Figure 30: Example use case *Performance Analytics*, showing the model architecture and workflow divided into *Model Layer*, *Module Layer* and *Component Layer* with their respective elements, including inputs and outputs per element (own figure).

Following the workflow outlined in Section 4.4, the initial step in the application of the PdM model involves a detailed description of the model's input and output attributes. This foundational process ensures a clear understanding and categorization of the data types and their roles within the model's architecture.

The input data for the model is categorized into two distinct types: `Raw Data` and `Meta Data`. `Raw Data` refers to the onboard logging information collected from the vehicle during the racing scenario, characterized by its high-frequency capture of multiple vehicle signals. Despite its comprehensive nature, this data is initially presented in an unspecified format, reflecting the raw and unprocessed state in which it is collected.

Conversely, `Meta Data` encompasses the expert knowledge that supplements the raw signal data. This metadata is crucial for interpreting the raw data's significance and context but is similarly provided in an unspecified format. The informal and implicit nature of this information, often communicated through direct communication channels, contributes to its unspecified format, calling for a flexible and interpretative approach to its integration within the model.

The model's output is designed to offer actionable insights into the feature importance for vehicle speed, articulated through two primary elements: the `Feature Importance List` and the `Evaluation Results`. Both elements are generated in formats that, while initially unspecified, are tailored to convey the model's findings effectively.

The `Feature Importance List` provides a ranked enumeration of vehicle signals based on their impact on performance, serving as a direct output of the model's analytical capabilities. Meanwhile, the `Evaluation Results` offer a comprehensive assessment of the model's accuracy and reliability, encapsulating the confidence level associated with the feature importance scores. Together, these outputs furnish a nuanced understanding of the model's predictive performance and its implications for PdM strategies.

This initial model description and data categorization set the stage for the subsequent analytical processes, ensuring that the model's application is grounded in a clear and structured understanding of both the input and output data dynamics.

Given the workflow mapped out in Section 4.4 and the unique characteristics of the example use case, it becomes evident that no existing model in the database adequately matches the required specifications. Consequently, it is essential to individually tailor each code module to align with the demands of the use case model. This customization process is guided by the defined Process Exit Points, ensuring a structured and coherent development approach.

The example use case, identified as *ML Model Insight Value Creation*, demands the integration of several distinct code modules to pave the way for the comprehensive analysis and interpretation of the data.

The `Data Input (DI)` module is essential in the initial phase of the PdM model's workflow, tasked with ingesting `Raw Data` and `Meta Data`. It converts the raw data into a structured data frame format, facilitating subsequent analysis and processing. Concurrently, it enriches the meta data by performing a descriptive statistical analysis of the input data, resulting in the creation of `Statistical Meta Data`. This enriched meta data provides a comprehensive statistical overview, enhancing the model's development process by ensuring that all data inputs are not only accurately captured but also effectively organized and analyzed.

The `Data Preprocessing (DP)` module is a critical component in the PdM model's pipeline, focusing on refining the input data to ensure its optimal condition for analysis. It takes `Input DF` and `Statistical Meta Data` as inputs and applies a series of transformations aimed at enhancing data quality. These transformations include addressing missing values, reducing noise, and filtering out irrelevant information. The result of this process is the `Preprocessed DF`, a cleaner and more analytically valuable dataset, ready for the subsequent stages of feature engineering and model development. This module underscores the importance of thorough data preparation in building effective PdM models.

The `Feature Creation (FC)` module generates new features from the `Preprocessed DF` and `Statistical Meta Data`. It focuses on the time series nature of this use cases data by sliding the data into windows and describing those windows with a multitude of numeric features. The outcome is the `Feature DF`, which contains these new features, tailored to enhance the model's analysis of complex patterns and relationships within the time series data.

The `Feature Labeling (FL)` module processes the `Feature DF` and utilizes `Statistical Meta Data` to assign labels to each feature. In the context of this use case, where each window corresponds to a single lap, the labeling process is directly linked to the window's duration, effectively representing the lap time. This methodical approach ensures that each feature within the `Feature DF` is accurately labeled, reflecting the time spent on each lap. The resulting `Labeled DF` comprises these thoroughly labeled features, thereby streamlining the identification and interpretation process within the model's analytical framework. This structured labeling mechanism is fundamental for the subsequent phases of model development, particularly in enhancing the model's predictive accuracy and interpretability.

The `Feature Encoding (FE)` module takes the `Labeled DF` as input and focuses on converting categorical features into a format that is compatible with the analytical model, optimizing its performance. Given that the signals in this use case are numeric, the encoding process primarily involves normalization. This step is essential for ensuring that all features contribute equally to the model's performance, preventing any single feature from disproportionately influencing the model due to its scale. The output of this process is the `Encoded DF`, which contains the normalized features, ready for MB and ME.

The `Model Building (MB)` module constructs the predictive model using the `Encoded DF` and `Statistical Meta Data` as inputs. This step is fundamental for developing the model that embodies the core analytical capabilities required for the use case. The construction of the model involves training it with the processed and encoded features to accurately predict outcomes based on the input data. The outputs of this module include the `Test DF`, which is used for evaluating the model's performance, the `Trained Model` itself, and the `Feature Importance List`. This list, highlighting the significance of each feature in the prediction process, serves as the first critical output requirement of the model, offering insights into which features most strongly influence the model's predictions.

The `Model Evaluation (ME)` module assesses the performance and reliability of the built model by utilizing the `Test DF`, `Statistical Meta Data`, and the `Trained Model` as inputs. This evaluation is vital for understanding the model's predictive accuracy and the confidence in its outputs. Through a comprehensive analysis, the module provides `Evaluation Results` that detail the model's effectiveness in making predictions. These results, constituting the second critical output requirement, offer valuable insights into the model's capabilities and limitations, thereby concluding the use case with a clear understanding of the model's performance and areas for potential improvement.

While the `Feature Visualization (FV)` module plays a vital role in presenting the model's findings in an interpretable and accessible manner, it is not further detailed within the context of this example use case for the sake of brevity and focus. The absence of functional Dp on the FV module in the initial stages of model development allows for this modularization, ensuring that the primary emphasis remains on the creation, evaluation, and optimization of the predictive model.

This modular approach to specifying the code components for the example use case underscores the flexibility and adaptability of the *Predictive Maintenance Framework*. By tailoring each module to the specific requirements of the use case, the framework allows a targeted and efficient development process, paving the way for the creation of robust and effective PdM models.

Given the absence of preexisting modules that fit the specific requirements of this use case, it is necessary to define components for each module within the workflow. While the workflow permits an organized approach by allowing the definition of a single component per module with direct copying of inputs and outputs, for enhanced clarity and understandability, certain modules are decomposed into multiple components. This decomposition permits a more granular understanding of the module's functionality and its contribution to the overall PdM model. By defining specific components for each module, the workflow ensures a comprehensive and tailored approach to model development, accommodating the unique aspects of the use case while maintaining the flexibility to adapt to the intricacies of PdM applications.

The `Data Input` Module consists of two main components. The first component, `Read File`, is tasked with ingesting `Raw Data` and converting it into a structured data frame, termed `Input DF`. This transformation is central for subsequent analysis and processing steps within the PdM model. Following this, the `Descriptive Statistics` component takes the `Input DF` and `Meta Data` to conduct a thorough statistical analysis. This process not only enriches the `Meta Data` with detailed insights about the data's characteristics but also produces `Statistical Meta Data`. This enriched `Meta Data` plays a vital role in informing further DP and analysis stages, ensuring a deep understanding of the underlying data patterns and distributions.

The `Data Preprocessing Module` enhances the `Input DF` through a sequence of targeted filters, each designed to improve data quality for the predictive model. The `Filter Constant Cols` component removes columns lacking variability, which is essential for the analysis. The process continues with `Filter by number of unique values`, leveraging `Statistical Meta Data` to discard features with less than `n_unique` values, thus focusing on more predictive elements. Similarly, `Filter by correlation values` aims to mitigate multicollinearity by filtering out features with correlation pairs exceeding `c_corr`. The `Filter Manual Outliers` and `Filter Statistical Outliers` components address outliers, the former through expert knowledge and the latter via statistical thresholds (`C_out`), ensuring data integrity. The culmination of this process is the `Apply Filters` component, which applies all filters to the `Input DF`, resulting in the `Preprocessed DF`, ready for further modeling steps.

The `Feature Creation Module` initiates with the `Define Windows by Value` component, which segments the `Preprocessed DF` into discrete windows based on criteria derived from `Statistical`

`Meta Data`. This segmentation process assigns a unique `Window ID` to each segment, effectively organizing the dataset for feature extraction. After that, the `Transform Windows to features` component takes the segmented `Preprocessed DF`, along with the `Window ID` and `Statistical Meta Data`, to generate new features. These features are then compiled into the `Feature DF`, with the `Window ID` serving as the index, helping the identification and analysis of features within the context of their respective windows.

The `Feature Labeling Module` begins with the `Label Target Feature` component, which utilizes the `Feature DF` and `Statistical Meta Data` to assign labels to the target feature. This labeling process is crucial for distinguishing the target variable in the dataset, allowing its identification and analysis in the model's learning process. The output is a `Column: Labeled Target`, which contains the labeled target feature.

Constitutively, the `Append Column` component integrates the `Column: Labeled Target` into the `Feature DF`. This step ensures that the labeled target feature is properly incorporated into the dataset, resulting in the `Labeled DF`. This dataframe, now with the target feature clearly labeled, is primed for the encoding and MB phases, ensuring clarity and consistency in the model's training and evaluation processes.

The `Feature Encoding Module` comprises the `Normalize Data Frame` component, which is responsible for the normalization of the `Feature DF`. This process adjusts the features within the dataframe to a common scale without distorting differences in the ranges of values, ensuring that each feature contributes equally to the analysis. The output of this component is the `Normalized DF`, ready for the subsequent modeling steps.

The `Model Building Module` features two critical components essential for developing the predictive model. The first, `Train Test Split`, takes the `Encoded DF` and `Statistical Meta Data` to partition the dataset into training and testing subsets. This separation is vital for evaluating the model's performance on unseen data, ensuring that the training and validation processes are robust and reliable. The outputs of this component are the `Train DF` and `Test DF`, which are used respectively for training the model and evaluating its performance.

The subsequent component, `Train Model`, utilizes the `Train DF` and `Statistical Meta Data` to construct and train the predictive model. This process involves fitting the model to the training data, allowing it to learn the underlying patterns and relationships. Upon completion, the output includes the `Trained Model`, ready for performance evaluation, and the `Feature Importance List`. This list provides insights into the relative importance of each feature in the model's predictions, highlighting the variables that play a vital role in the model's decision-making process.

The `Model Evaluation Module` is economized into a single, central component: `Evaluate Results`. This component leverages the `Test DF`, `Statistical Meta Data`, and the `Trained Model` to assess the model's performance. The evaluation process encompasses a thorough analysis of the model's predictive accuracy and reliability, utilizing the test dataset to measure how well the model generalizes to new, unseen data. The outcome of this evaluation is encapsulated in the `Evaluation Results`, which detail the model's effectiveness and provide a comprehensive overview of its predictive capabilities. These results are instrumental in validating the model's utility and guiding any necessary adjustments or optimizations to enhance its performance.

4.5.2 Refinement of Attributes

In this work, the exploration extends into a distinct scenario to examine the attributes system's adaptability and robustness further. The focus is on advanced pattern recognition, aiming to develop a trained classifier as the model output. The complexity and nature of the `Raw Data` introduce new challenges and refinement steps in the model development process. I4.0 ushers in sophisticated ML techniques that significantly bolster the efficacy of PdM strategies by accelerating automated fault detection and diagnosis, thereby minimizing downtime and enhancing the utilization rate of components. This paradigm shift underscores the indispensability of PdM in achieving sustainable smart manufacturing in I4.0 (Çınar et al., 2020). Unlike the initial use case, the `Raw Data` in this scenario is characterized by a different format, demanding a tailored approach to preprocessing and feature extraction. This diversity in data format accentuates the need for a flexible attributes system capable of accommodating varying data characteristics while preserving the integrity and effectiveness of the PdM model.

Furthermore, the model output, centered around pattern recognition, calls for an `Evaluation Results` phase that not only confirms the model's accuracy but also furnishes interpretative insights. This phase is central for ensuring the model's real-world applicability and for its subsequent refinement based on empirical evidence. A defining aspect of this use case is the non-trivial window size for feature generation, which plays a crucial role in the model's capacity to discern pertinent patterns within the data. Opting for an appropriate window size is essential, as it directly impacts the model's ability to balance the inclusion of adequate contextual information against the risk of signal dilution.

The complexity ushered in by the choice of window size exemplifies the iterative nature of model development in PdM applications. It emphasizes the need for a dynamic and iterative approach to refining the attributes system, ensuring its responsiveness to the specific demands of each use case. The introduction of this sophisticated pattern recognition use case illuminates several implications for the attributes system. The system must adeptly manage varying data formats, ensuring consistent integration and preprocessing of raw data inputs. Additionally, the attributes system should furnish mechanisms to tailor the feature extraction process, particularly in determining the optimal window size for diverse scenarios. Moreover, to cater to the model output's focus on pattern recognition, the system requires robust evaluation metrics capable of validating and interpreting the results effectively. These considerations highlight the prerequisite for a comprehensive and adaptable attributes system, equipped to meet the evolving requisites of PdM model development. Through iterative refinement and a focus on the specific challenges posed by each use case, the system can significantly augment the development efficiency and usefulness of PdM applications (Davari et al., 2021; Ucar, Karakose, and Kırımça, 2024).

In the transition from the initial use case to the subsequent one, a critical evaluation of the `Data Input Module` reveals significant alterations necessitated by the shift in the nature of `Raw Data`. Initially, the data is encapsulated in a `.csv` file format, a widely recognized structure for tabular data due to its simplicity and broad compatibility with a range of data analysis tools. The component designed for reading this data, aptly named the `Read Data` Component, is thus optimized for parsing `.csv` files, enabling efficient data processing and preparation for model development tasks.

However, the evolution into the second use case introduces a vital change; the raw data is now presented in a `.json` format. The `.json` format, characterized by its hierarchical structure of key-value pairs, offers enhanced flexibility for representing complex data relationships and nested arrays, making it particularly suitable for data embodying varied levels of complexity. This transition to a `.json` format calls for a reevaluation of the `Read Data` Component, as the techniques employed for `.csv` file ingestion are inherently unsuitable for the `.json` data structure. Reading `.json` files demands parsing strategies capable of interpreting their nested formats, extracting pertinent information whilst maintaining the integrity of the hierarchical data relationships.

This demands a redesign or significant adaptation of the `Data Input Module`, underlining the importance of versatility in handling diverse data formats effectively. It also highlights the need for PdM systems to incorporate flexible, format-agnostic data ingestion capabilities to accommodate the variability inherent in raw data formats across different use cases.

To address this adaptation, a modular approach in the system's architecture is advocated, allowing for the integration of multiple data reading components, each tailored to a specific data format. This ensures the system's preparedness for diverse data types, promoting a smoother transition between use cases and enhancing the system's overall robustness and applicability in varied PdM scenarios.

The detailed examination of the transition between use cases, starting with the `Data Input Module`, illustrates the elaborate Dp between raw data characteristics and the system's data processing capabilities. It serves as a quintessential example of the necessity for adaptability in the design of PdM systems, ensuring their usefulness across a multitude of application contexts.

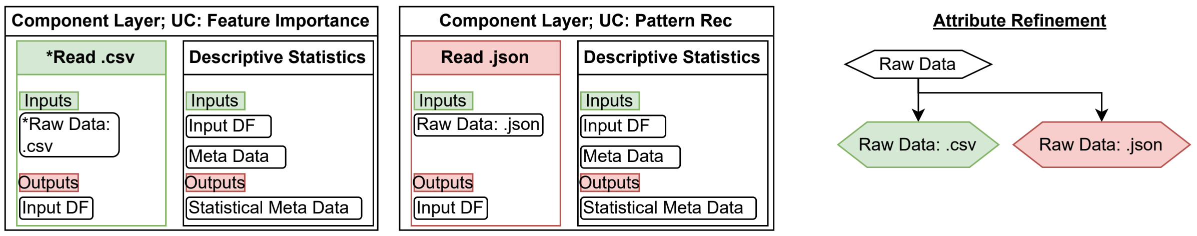


Figure 31: Updating descriptive attributes: *Data Input*. Left: Updated component layer for the use case *Feature Importance* and new component layer for the use case *Pattern Rec*. Right: Hierarchical attribute refinement with hierarchical tree graph of component attributes (own figure).

In Figure 31, the updated component layer for the initial use case alongside the new component layer for the current use case is depicted. Initially, the `Read File` component, designated for parsing `.csv` files, is refined to `Read File: .csv` with a modified input attribute to explicitly indicate `Raw Data: .csv`. To accommodate the new use case involving `.json` formatted data, a distinct `Read File: .json` component is introduced. This new component, while mirroring the output attributes of its predecessor, ensures compatibility within the `Data Input` Module for processing differing data formats, thereby maintaining the module's versatility.

The Hierarchical Attribute Refinement, illustrated on the right side of Figure 31, showcases the hierarchical tree graph of component attributes. It is evident from the graph that the newly specified child attribute `Read File: .json` cannot simply be added to the parent attribute `Read File: .csv`. Instead, a sibling attribute `Read File: .csv` must be mapped out to respect the inapplicability of the predefined component attributes for handling `.json` files. This structural refinement underscores the necessity for a systematic approach to attribute specification within the system's architecture, ensuring that each component is accurately represented and functionally coherent within the context of its intended use case.

The hierarchical relationship between attributes in the system's architecture is effectively managed through the application of namespaces, a practice that simplifies the organization of system components. For example, using namespaces such as `Read File: .csv` and `Read File: .json` allows for a clear distinction between components designed for reading `.csv` and `.json` formatted data, respectively. This naming convention not only aids in avoiding naming conflicts but also enhances the system's modularity and scalability. As such, namespaces play an important role in the systematic organization of attributes, ensuring that each component's functionality and scope are accurately represented within the system's overall architecture.

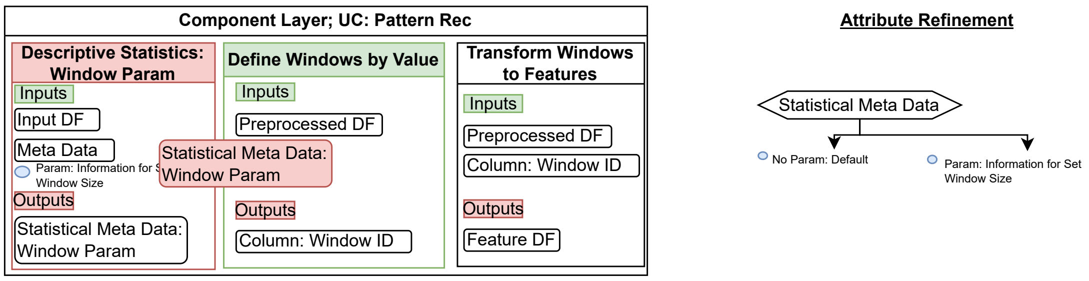

Component Layer; UC: Pattern Rec
Descriptive Statistics: Window Param
Inputs
Input DF
Meta Data
Param: Information for Set Window Size
Outputs
Statistical Meta Data: Window Param
Statistical Meta Data: Window Param
Define Windows by Value
Inputs
Preprocessed DF
Outputs
Column: Window ID
Transform Windows to Features
Inputs
Preprocessed DF
Column: Window ID
Outputs
Feature DF
Attribute Refinement
Statistical Meta Data
No Param: Default
Param: Information for Set Window Size

Figure 32: Updating descriptive attributes: Feature creation. Left: Introduction of novel component window param. Right: Hierarchical attribute refinement with hierarchical tree graph of component attributes (own figure).

In the progression of refining the FC module for the PdM system, a critical adaptation is introduced to manage the specification of window size—a key parameter in the transformation of data windows into features. This refinement is essential for the `Transform Windows to Features` component, as the approach to defining window size significantly diverges from the methodology employed in the initial use case. Illustrated in Figure 32, this step addresses the challenge wherein the window size cannot be inferred directly from a column value, nor is it included in the `Statistical Meta Data` generated by the `Data Input` Module.

To accommodate this requirement without altering the existing workflow of the initial use case, a novel component—termed `Descriptive Statistics: Window Param`—is introduced. This component is designed to augment the `Statistical Meta Data` with an optional window size parameter. The flexibility of this design lies in its ability to overwrite the `Window ID` column when necessary, ensuring compatibility across use cases with varying data structuring needs.

The incorporation of the `Descriptive Statistics: Window Param` component into the system architecture represents a nuanced approach to enhancing the `Statistical Meta Data` without disrupting the component's applicability in the initial use case scenario. This is achieved by introducing an optional parameter to the metadata creation process, thereby allowing for the dynamic adjustment of the `Window ID` based on the specific requirements of the current use case. As depicted in Figure 32, this refinement enables the `Define Windows by Value` component to utilize the newly added parameter, adapting its output accordingly. In scenarios where the window size parameter is absent, the component retains its default behavior, mirroring the process outlined in the initial use case and ensuring a consistent integration into the FC module.

This strategic enhancement to the FC module underlines the importance of flexibility and adaptability in the development of PdM systems, especially when dealing with the complexities of data transformation and feature extraction. By allowing for the optional specification of window size, the system can effectively cater to a broader spectrum of use cases, each with unique data characteristics and analysis requirements, thereby ensuring a more robust and versatile PdM solution.

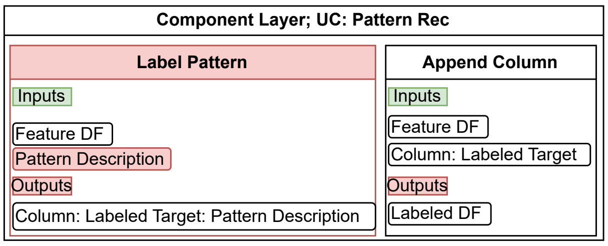


Figure 33: Updating descriptive attributes: Feature labeling with introduction of label pattern component to replace the label target feature component (own figure).

In the revised FL Module, significant alterations are implemented to accommodate a new use case. Unlike the initial scenario, where the label is determined by the *time spent per lap* and the model is a regression model focused on predicting lap times, the current use case shifts towards a classification model designed to predict the occurrence of certain patterns. Consequently, the labeling module now incorporates a `label column` generation process, predicated on a `Pattern Description`. This `pattern description` serves as an additional model input and encapsulates a set of criteria that must be met within a specified data window.

As illustrated in Figure 33, the `Label Pattern` component is introduced to replace the `Label Target Feature` component entirely. This modification demands two primary inputs: the `Feature DF` (Feature Data Frame) and the `Pattern Description`. The output is a thoroughly generated label column, which seamlessly integrates with the existing `Append Column` component to ensure the module's functionality remains intact.

The `Feature Encoding` Module is significantly adapted to accommodate the inclusion of both

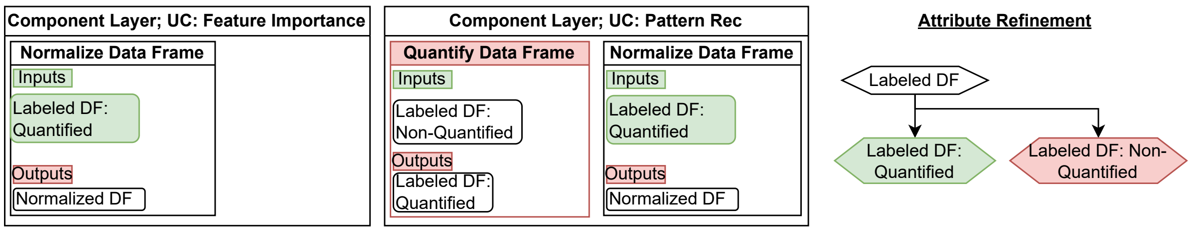


Figure 34: Updating descriptive attributes: Feature creation. Left: Integration of additional component quantify data frame and classification of Labeled DF into quantified and non-quantified. Right: Hierarchical attribute refinement with hierarchical tree graph of component attributes (own figure).

quantitative and non-quantitative (text-based) data in the new raw data set. This diversification of data types requires the implementation of machine-readable encoding techniques, such as one-hot-encoding, to ensure that non-quantitative data is appropriately processed for model training. Unlike the initial use case, which exclusively dealt with quantitative features and did not require this encoding step, the current approach demands a more sophisticated data preparation method.

As depicted in Figure 34, this modification leads to the integration of an additional component, `Quantify Data Frame`, into the module's component layer. This component is responsible for transforming text-based data into a format that can be effectively utilized by ML algorithms. Furthermore, the attributes of the `Labeled DF` are now distinctly classified into two categories: `Labeled DF: Quantified` and `Labeled DF: Non-Quantified`. This specification allows for a clear demarcation between data types and advances the targeted processing of each.

The `Labeled DF: Quantified` attribute, in particular, is refined to serve as a prerequisite for the reusability of the pre-existing `Normalize Data Frame` component. This refinement underscores the module's enhanced capability to handle a broader spectrum of data types, thereby extending the flexibility and applicability of the encoding process to support the varied requirements of the new use case.

The adaptation of the input and output attributes to accommodate diverse scenarios underscores the system's flexibility and the iterative nature of its evolution. This process is articulated through four principal refinement categories, each addressing different dimensions of system modifications to meet emerging requirements:

1. **Specification of Component Functionality**: This involves the precise outline of component functionalities, required by variations in the nature of `Raw Data`. An example is the transition from processing `.csv` files to `.json` files, which required a reevaluation and specification of the `Read Data` component to ensure effective parsing of `.json`'s hierarchical

structure. This adaptation highlights the critical role of specific functionality definition in efficiently managing diverse data formats.

2. **Additional Functionality with Added Parameters and Default Behaviour**: Components are adapted or newly introduced with additional parameters to extend their functionality, maintaining compatibility with the existing workflow. For instance, the inclusion of the `Descriptive Statistics: Window Param` component in the FC module introduces an optional window size parameter, offering flexibility across use cases while preserving default behavior when the parameter is not utilized.
3. **Replacement of Components and Integration of New Model Inputs**: This category covers the replacement or integration of components to accommodate different modeling approaches or data processing requirements. The replacement of the `Label Target Feature` component with the `Label Pattern` component, for instance, signifies a shift towards classification models and calls for new inputs like the `pattern description` for appropriate label generation.
4. **Refinement of Input Attributes for Components**: Focuses on the refinement of component input attributes to process a broader range of data types effectively. An example is differentiating the `Labeled DF` into `Quantified` and `Non-Quantified` categories, fundamental for the `Feature Encoding` Module's ability to handle both quantitative and non-quantitative data, highlighting the necessity for an adaptable attribute system.

These refinements collectively enhance the attribute system's robustness and adaptability, ensuring its applicability across various PdM scenarios through specific component functionality specifications, the integration of additional parameters with default behaviors, the replacement of components for new model inputs, and the refinement of input attributes.

4.5.3 Overview of the Attributes System

In the workflow presented in this work, initiating and refining models, modules and components are critical steps that ensure the quality and reusability of the objects. The initiation step involves setting up the basic structure and parameters of the model based on a predefined set of attributes that describe the essential characteristics of the data and the system. Refinement, on the other hand, involves iterative adjustments to the model based on feedback from its performance and changing operational conditions. This adaptive approach helps in targeting specific areas of the model that require more detailed analysis and optimization.

A detailed description of every component in a PdM system would indeed be inefficient due to the vast quantity of data and the complexity of interactions among system components. This is supported by literature indicating that overly detailed models can lead to excessive computational costs and diminished practical usability due to their complexity. These sources argue for the balance between detail and efficiency, emphasizing that not all system aspects require the same level of examination.

The iterative system proposed in this work aligns with efficient PdM practices by enabling focused efforts on areas where a higher level of detail is necessary. This is achieved by initially applying a standard level of detail to all components and then selectively refining those components based on their performance metrics and the critical nature of their operation within the system. Such a targeted approach ensures that resources are allocated efficiently, enhancing the system's overall effectiveness without the unnecessary overhead of detailed analysis for every component.

This methodology not only conserves resources but also enhances the model's responsiveness and adaptability, which are crucial in dynamic operational environments typical of industrial applications. By implementing an iterative and targeted refinement process, the model remains agile, with adjustments made swiftly to cater to emerging needs or to address areas showing potential for optimization.

This work introduces a formal attribute system, effectively bridging a critical research gap by enabling the precise and dynamic comparison of pipeline objects within the *Predictive Maintenance Framework*. Drawing from the principles of the *Z* Specification language and incorporating the flexibility of modern automated code summarization and data description systems, this system enhances the functional assessment and comparability of pipeline components. By aligning with the agile and modular nature of the proposed PdM processes, it significantly augments traditional methods, offering a scalable and efficient solution tailored to the evolving demands of industrial applications.

4.6 Conclusion of the Results

This chapter concludes with a synthesis of the execution and validation of the artifacts developed in this work, namely the *Use Case Description Methodology*, *Component Repository* with interface standardization and a priori analysis, *Component Creation Workflow*, and the *Descriptive Attributes System*. These artifacts collectively forge a robust framework for enhancing the development efficiency of PdM models and AI methodologies. The implementation of these tools not only supports a systematic development process but also significantly augments the efficiency and effectiveness of model development within a high-interest industrial context.

The *Use Case Description Methodology* provides a structured approach to capturing and analyzing requirements, ensuring that all potential use cases are comprehensively addressed. The *Component Repository*, enhanced with interface standardization and a priori analysis, offers a centralized platform that facilitates the reuse of components, thereby reducing redundancy and accelerating development timelines. The *Component Creation Workflow* establishes a procedural roadmap for developers, guiding them through the stages of component development with a focus on consistency and quality control. Lastly, the *Descriptive Attributes System* plays an essential role in defining and categorizing the properties and functionalities of components, which enhances discoverability and usability within the repository.

Together, these artifacts create a complex and rich platform that not only sets the stage for future research into PdM model development and AI development methodologies but also realizes significant efficiency gains in practice. Stand-alone, yet integratively robust, these tools cover the full spectrum of the development process, from conceptualization to deployment. Backed by empirical evidence and thorough evaluation in a demanding industrial environment, these innovations hold substantial credibility and promise substantial impacts on the field. The integration of these systems into the development lifecycle is poised to provide a transformative influence on the efficiency and effectiveness of PdM systems and AI applications, marking a significant advancement in the domain of software engineering for predictive technologies.

Chapter 5

Discussion and Outlook

This final chapter aims to synthesize the insights and findings presented throughout this thesis, providing a comprehensive overview of the implications, limitations, and future directions of this research. The development and integration of the *Use Case Description Methodology*, *Component Repository* with interface standardization and a priori analysis, *Component Creation Workflow*, and the *Descriptive Attributes System* are each detailed in previous chapters, emphasizing their roles in enhancing the efficiency and effectiveness of PdM model development and AI development methodologies. Here, these elements are discussed in the context of their practical applications, theoretical contributions, and potential for future innovation.

5.1 Discussion of the Results and the Implications of the Key Findings

This section looks into the critical analysis of the key findings derived from this work, which seeks to enhance the understanding and implementation of PdM systems within industrial applications. Initially, a detailed exposition of the key findings is presented, highlighting how each contributes significantly to the research questions of this work. These findings not only answer the posed research questions but also validate the effectiveness and efficiency of the developed artifacts. By systematically presenting these findings, this discussion aims to provide a comprehensive overview of the implications and practical applications of this research in advancing PdM methodologies and technologies.

5.1.1 Overview of the Key Findings

The key findings of this work, highlighting the vital advancements and insights gained from the development and evaluation of the proposed frameworks and methodologies are discussed in the following.

Key Finding 1: Importance and Growth of Predictive Maintenance in the Automotive Industry

The first significant finding of this work underscores the growing role of PdM in industrial applications, particularly within the automotive sector. As industries progressively acknowledge the criticality of maintenance operations, PdM evolves not only in complexity but also in its value across various fields. This is especially prominent in the automotive industry with an exponential increase in the deployment of PdM applications. The integration of PdM systems in this sector is driven by the need to enhance operational reliability and efficiency, reduce downtime, and cut maintenance costs. This trend is indicative of a broader shift towards more sophisticated, data-driven maintenance strategies that leverage real-time data analytics and ML to predict and prevent potential failures before they occur.

Based on the current scientific literature, the importance and growth of PdM in the automotive industry can be highlighted by its effectiveness in improving operational efficiencies and reducing costs. Research shows that PdM, by utilizing advanced data analytics and AI, can significantly reduce maintenance costs and minimize unplanned downtimes in the automotive sector. For instance, studies illustrate that PdM systems enhance the lifespan of vehicles, lower the frequency of maintenance, and can predict potential failures more effectively than traditional maintenance approaches.

The integration of PdM within the automotive industry demonstrates significant operational efficiencies and cost reductions. Studies indicate that by leveraging advanced data analytics and AI, PdM significantly curtails maintenance costs and reduces unplanned downtimes. For instance, PdM strategies are shown to enhance vehicle lifespan and decrease maintenance frequency, effectively predicting potential failures and thereby optimizing maintenance schedules (Arena et al., 2021; Ucar, Karakose, and Kırımça, 2024). These advancements not only yield direct economic benefits but also enhance vehicle safety and reliability, critical factors in the automotive sector. Furthermore, the deployment of ML models for PdM in automotive engine components emphasizes the capability of these technologies in early fault detection. This predictive capability is fundamental for timely maintenance interventions that prevent more severe issues, thereby underlining the transformative impact of PdM from a reactive to a predictive, data-driven maintenance approach (Tessaro, Mariani, and Coelho, 2020).

Key Finding 2: Research Gap in Methodologies for Predictive Maintenance System Development

This work identifies a significant research gap in the methodologies employed to enhance the development of PdM systems. While established software development domains successfully incorporate CBSE methods tailored for industrial applications, PdM development does not yet achieve a similar integration, particularly in contexts involving data science and industrial applications. The lack of established methodologies that cover the full spectrum of model development within the PdM domain underscores a critical need for innovation in this area. The integration of data science techniques into industrial PdM systems remains emergent, with considerable potential for developing standardized methodologies that could significantly improve the usefulness and efficiency of PdM systems (G. Li et al., 2021; Nunes, Santos, and Rocha, 2023).

Key Finding 3: The Role of Component-Based Software Engineering in Advancing Predictive Maintenance

The third important insight from this research highlights the significance of CBSE in elevating the technology-readiness-level for PdM systems. CBSE emerges as a vital methodology in refining and accelerating the development process for PdM technologies, particularly by promoting modular and reusable components. However, for CBSE to fully deliver its benefits, certain criteria must be met within the application domain. These include compatibility with existing systems, scalability to meet different industrial needs, and adaptability to evolving technologies. The application of CBSE in PdM underscores a need for a structured approach that not only adheres to these criteria but also leverages them to enhance system integration and operational efficiency. Studies indicate that when these conditions are satisfied, CBSE can significantly rationalize the development process, reduce time-to-market, and increase the robustness of PdM systems.

Key Finding 4: Applicability of CBSE Principles in PdM Model Development

This work's first research artifact, the *Predictive Maintenance Framework*, substantiates the effectiveness of integrating CBSE principles into the development of PdM models. This finding is supported by recent studies indicating that employing CBSE methodologies enhances both the scalability and robustness of PdM systems. These systems are increasingly reliant on complex data streams and require a modular approach to handle diverse industrial applications effectively. The use of CBSE in PdM allows for a systematic development process where components can be developed, tested, and validated independently before their integration, ensuring higher quality and reliability of the PdM systems. This approach not only aligns with the current trends in software engineering but also meets the specific demands of PdM in varied industrial contexts (Ucar, Karakose, and Kırımça, 2024; Arena et al., 2021).

Key Finding 5: Validating the Predictive Maintenance Framework through Individual Use Case Analysis

This work's second artifact demonstrates how describing individual PdM use cases not only highlights the complexity and multifaceted nature of PdM but also reveals an inherent structure within these applications. This finding further validates the PdM Framework developed in this research. The literature supports the use of a component-based approach for developing PdM models, noting that this method enhances the system's adaptability and effectiveness across various industrial applications. By leveraging insights from the literature, particularly those that discuss the integration of AI and ML within PdM strategies, this artifact lays a foundational framework that supports the component-based development of PdM models. This approach aligns with the principles of I4.0, where PdM plays a crucial role in sustainable manufacturing by minimizing downtime and optimizing the use of equipment (Achouch et al., 2022).

Key Finding 6: The Significance of Component Repository Structure in PdM Systems Development

The sixth key finding from this research highlights the central role of the Component Repository Structure as a core artifact in the development of PdM systems. This structure not only consolidates the insights gathered from previous artifacts but also introduces a set of component design rules centered around a priori reusability. The literature suggests that adopting a component-based approach significantly aids in handling the complexities involved in PdM systems by allowing components to be reused across different scenarios, which enhances the efficiency and scalability of the maintenance processes. Moreover, the structured repository ensures that each component is optimized for reuse and can be easily integrated or modified to meet specific PdM requirements, thereby supporting sustainable and efficient PdM practices in I4.0 settings (Lv et al., 2023; C.-H. Chen et al., 2017).

Key Finding 7: Advantages of the Component Creation Workflow in PdM Systems Development

The seventh key finding of this research reveals that the component creation workflow, despite its initial higher costs, offers significant advantages over traditional approaches in the management of PdM systems, particularly in areas of high complexity. The implementation of this workflow in PdM systems development ultimately achieves a break-even point through reduced development costs over time. Studies show that while the upfront investment in PdM can be substantial, the long-term benefits include reductions in maintenance costs and operational disruptions, thereby justifying the initial expenditure. The use of advanced predictive models and continuous improvement strategies, as part of the component creation workflow, enhances the efficiency and reliability of maintenance operations, contributing to overall cost savings and improved asset management.

Key Finding 8: Significance of the Descriptive Attributes System in PdM Component Workflow

The eighth key finding of this research emphasizes the vital role of the descriptive attributes system within the component creation workflow for PdM systems. This system is not only vital in streamlining the development process but also offers substantial potential for further enhancements and refinements. The implementation of a detailed descriptive attributes system allows for a more structured and efficient design and deployment of PdM components by providing clear, actionable data that informs development decisions. Research indicates that such systems help optimize maintenance strategies by improving the precision and efficiency of PdM tasks, ultimately leading to cost savings and enhanced equipment reliability.

5.1.2 Reflection on Research Questions

With the overview of this work's key findings, a discussion of the comprehensive examination of the research questions posed at the outset of this investigation is conducted. Each research question is thoroughly addressed through the development and analysis of specific artifacts created as part of this work. The artifacts, each designed with a distinct focus within the realm of PdM, not only answer the fundamental questions but also contribute to advancing the field. This structured inquiry not only substantiates the proposed PdM Framework but also highlights its adaptability and usefulness across different industrial applications, thus providing a robust foundation for future research and development in PdM systems.

RQ1: Is it possible to classify the development of a PdM application in a framework?

The initial research question inquires whether it is possible to classify the development of a PdM application within a structured framework. The *Predictive Maintenance Framework*, developed as the primary artifact of this study, lays a comprehensive literature foundation that reveals a consistent structure across PdM models documented in related scientific literature. This framework successfully captures and categorizes the inherent similarities across these models, demonstrating its robustness and applicability. Furthermore, the validity of this framework is reinforced in real-world settings, as observed in the deployment environments examined in this research. This is further substantiated by the development and application of the second artifact, the Use Case Description Methodology. Together, these elements confirm that the development of PdM applications can indeed be systematically classified within a framework, thereby affirmatively answering RQ1. This finding not only enhances the understanding of PdM system development but also guides future research and practical implementations in this domain.

RQ2: Is it possible to design a component repository using the similarities while respecting the individual aspects of PdM?

The second research question explores the feasibility of designing a component repository that leverages the similarities across PdM systems while respecting their individual aspects. The development and evaluation of the third artifact, the Component Repository Structure, provide a concrete answer to this question. This repository successfully implements CBSE principles within the PdM domain, demonstrating that components can be standardized without requiring changes to the content, thus preserving the quality of the PdM models. Additionally, the repository utilizes design principles derived from reusability measurements, which lay a robust foundation for the repository's efficient expansion across an increasing number of PdM use cases. These findings affirmatively answer RQ2 by showing that it is indeed possible to design a component repository that maintains the integrity and specificity of PdM applications while fostering standardization and reuse.

RQ3: Is it possible to reduce development costs for PdM applications?

The third research question investigates whether it is possible to reduce the development costs for PdM applications. Through the development and subsequent validation of the Component Creation Workflow, this research demonstrates a significant reduction in development costs. The application of event-discrete simulation and sensitivity analysis within this workflow enables a detailed examination of the factors influencing cost reduction. Results indicate that the reusability costs in the system predominantly hinge on the effective description of model objects. The final artifact, the Descriptive Attributes System, proves instrumental in achieving cost-efficiency early in the integration process and maintaining a low percentage of additional reusability costs. This robust approach substantiates the assertion that development costs for PdM applications can indeed be reduced, confirming a positive answer to RQ3 and illustrating effective strategies to optimize economic efficiency in PdM systems development.

The comprehensive investigation and development of the artifacts within this work substantively and beneficially answer the posed research questions, thereby offering a novel methodology for the development of PdM models. These artifacts not only underscore the viability of classifying PdM development within a structured framework, designing a reusable component repository, and reducing development costs, but also significantly enhance the technological readiness level of the field. This research lays a foundational step for future explorations and provides a robust platform that can be built upon by subsequent studies in a multitude of facets. The implications of this work are vast, promising broad applications across various industries, thereby marking a significant contribution to the body of knowledge in PdM systems development.

5.2 Analysis of the Limitations of this Work

In this work, the limitations of the research artifacts are examined to ensure a comprehensive understanding of their constraints and implications. The analysis of each artifact adheres to a structured approach, outlining limitations into four key categories: methodological, theoretical, data-related, and practical constraints.

For each artifact, the discussion begins with *Constraints*, identifying and categorizing the inherent limitations under methodological, which relates to the design and procedural choices made; theoretical, addressing the underlying assumptions and conceptual frameworks; data-related, concerning the quality, sufficiency, and handling of data; and practical, which includes logistical, technological, or resource-based challenges encountered during the research. Following the constraints analysis, the *Implications* of these limitations on the reliability, validity, and applicability of the research findings are elaborated. The section is rounded off with *Learnings*, reflecting on the adjustments made and insights gained, which are instrumental for refining future research methodologies and strategies.

Limitations: Predictive Maintenance Framework

The methodological constraints of this work stem from its reliance solely on scientific literature. As is common in industrial research, it is challenging to ascertain the internal practices of companies from published sources alone, since many operational details remain proprietary and undisclosed.

A theoretical constraint is the lack of causal connections specifically within the PdM domain. The framework's structure is derived from the scientific adaptation of PdM use cases and a general scientific approach to problem-solving, rather than direct empirical evidence from the PdM field.

Data-related constraints include the inability to determine from the available data whether the framework's building blocks reflect an equitable distribution of development effort. If the majority of the development effort is concentrated on components without internal structure, this could significantly skew the effectiveness and focus of the framework.

Practical constraints involve the framework reflecting a functional level of structure, which may not accurately represent the division of problems within the industry. It is unclear whether the division is biased by the demonstration of the problem-solving approach or if it represents a functional split.

The implications of these constraints necessitate focusing on validation through observation within the research environment used in this study. Although the separation of development and evaluation stages led to successful validation, this success is limited to a relatively broad setting. A comprehensive validation process is required to confirm these findings across different environments.

The key reflection from addressing these limitations is that even with an extensive database, the applied method can only provide reliable results when combined with specific industry applications. The framework alone is insufficient to define the structure of industrial environments comprehensively.

Limitations: Use Case Description Methodology

A significant methodological constraint arises from the predetermined, non-negotiable structure of the canvas used in this study. This fixed structure limits the assessment of completeness across various potential use cases, validated only within the specific research environment where it was developed.

In terms of data-related constraints, the complexity and interfaces characteristic of industrial environments, such as those in this work, may exceed standard expectations. This discrepancy can disproportionately influence the perceived importance of different aspects of the artifact canvas.

Practically, the methodology faces limitations in scope due to its requirement not only to analyze content but also to adapt to different process environments effectively.

These constraints suggest that although the artifact is validated positively within the extensive, yet specific confines of this work's environment, numerous potentially valuable applications remain unexplored and untested.

The main takeaway is that the canvas approach effectively ensures the completeness of factors considered and eases evaluation. However, its utility as an environment-independent tool depends critically on insights gained from subsequent artifacts, highlighting its conditional applicability.

Limitations: Component Repository Structure

A key methodological constraint is the focus on the DE aspects, which are predominantly use case and data source centric. However, a comprehensive coverage of data infrastructure is not established, limiting the breadth of the repository's applicability.

Theoretically, the component repository presupposes a fairly even distribution of development resources across various modules. Challenges arise if use cases from different fields demand disparate resource allocations per module or require integration of modules with specific technical constraints on their interfaces. Such discrepancies call for further investigation into the repository's application across diverse scenarios.

The primary data-related constraint lies in the validation scope, which is restricted to insights from existing literature and use cases specific to the current research environment. Consequently, a generalized approach is not yet developed.

Practically, the effectiveness of a shared repository also hinges on robust version control and collaboration tools, aspects that are not addressed in this framework. Furthermore, there is a prerequisite for training all participants in the use of these tools to ensure smooth operational integration.

These constraints imply that, despite positive evaluations, it is crucial to assess environmental aspects and constraints thoroughly before the repository's findings can be implemented reliably.

The learnings highlight the importance of limiting the application of this artifact to environments where the interfaces and operational conditions closely match those encountered in this work's research setting. This is essential to prevent the need for additional, unforeseen research adjustments.

Limitations: Component Creation Workflow

A significant methodological constraint involves the challenge of validating with real data and calibrating parameters accordingly. This process is essential but often constrained by the availability and quality of data in real-world settings.

Theoretically, the limitation is that discussions on the simulation model artifacts are confined to those defined within the project scope. To enhance the validity of these models, calibration with real data in an implemented system is essential but is not accomplished within this work.

For simulation models, especially in sensitivity analysis, the key data-related constraint is the balance between computational resources and the need for comprehensive data handling. It is key to identify the most efficient use of given computational resources to compensate for any data-related limitations.

These constraints underscore that while the parameter results validate the hypotheses of this work and align with observations, a more detailed simulation analysis is necessary for a thorough analysis of system-wide implications.

The primary learning from this process is the value of sensitivity analysis as a potent tool in workflow simulations. It enables the evaluation of theoretical models even in the absence of extensive data, demonstrating its utility in testing hypotheses under constrained conditions.

Limitations: Descriptive Attributes System

A significant methodological constraint lies in the benchmarking of this system against complex problems, which calls for rigorous validation. Due to the inherent complexity, the application field in this work is narrowly defined, restricting broader generalization.

Theoretically, the system is limited by the absence of a comprehensive framework that can predict the performance impacts of various descriptive attributes in diverse scenarios. This lack of theoretical underpinning constrains the predictability and adaptability of the system across different contexts.

Given the large number of iterations required for validation, it is impractical to fully and quantitatively measure the system's effect in advance. This data-related limitation affects the reliability of predictive outcomes prior to real-world application.

Practically, the system's deployment is constrained by the availability of adequate computational resources and the need for continuous data updates to maintain its relevance and accuracy in dynamic environments.

These limitations imply that while this artifact shows promise based on qualitative evaluations, its implementation in real-world settings must be preceded by extensive validation with real-world data. Only through such stringent testing can the effectiveness and robustness of the system be adequately assessed.

The implications from this work highlight significant gaps in current research, particularly in understanding the impact of descriptive attributes on system performance. Although this work provides a qualitatively positive evaluated solution, it underscores the necessity for further empirical studies to bridge these gaps and refine the system for practical application.
As this detailed exploration of limitations concludes, this work recognizes that while the findings from various research artifacts are thoroughly validated within specific contexts, they are inherently bound by methodological, theoretical, data-related, and practical constraints. These limitations emphasize the necessity for a broader validation and adaptation across diverse environments, which is fundamental for the generalizability and robustness of the results. These insights pave the way for the next section of this work, Outlook, where the future directions for research will be outlined. This upcoming section aims to address the gaps identified in this analysis by proposing enhancements and exploring new avenues for applying these research artifacts in broader and more varied industrial contexts.

5.3 Outlook and Recommendations for Future Research

This segment of the thesis outlines strategic directions for future research, focusing on addressing the limitations identified in the previous sections. The intent is to enhance methodological approaches, expand theoretical frameworks, improve data management techniques, and refine practical implementations of PdM systems. Each proposed step is aimed at overcoming specific challenges, thereby advancing the field of PdM within various industrial applications.
The analysis highlights significant methodological limitations, particularly the reliance on specific data sets and environments which may not universally represent industrial practices. Future research should explore the incorporation of more diverse data sources, including real-time data from a broader spectrum of industrial settings. Additionally, the adoption of mixed methods research could enhance the depth of understanding and reliability of PdM frameworks. Enhancing simulation models to include adaptive learning algorithms might also help in forecasting and mitigating potential system failures in untested scenarios.
The limitations noted in the theoretical constructs of the current frameworks suggest a need for a more robust theoretical underpinning. Future work should focus on developing new or refining existing theories to better accommodate the variability and complexity of real-world industrial environments. Collaborative research initiatives with academic institutions could be beneficial in accurately testing these theories, leading to more refined models that are capable of handling the

complicated dynamics of PdM.

Data-related constraints identified include the limitations in the scope and scalability of data analysis methods currently used. To address this, it is proposed that future projects invest in the development of advanced analytical tools that can handle larger datasets more efficiently and with greater accuracy. Implementing more comprehensive data validation techniques to ensure the accuracy and reliability of the data used is also critical. Moreover, creating guidelines for data management that standardize data collection, storage, and analysis processes across industries could greatly enhance the scalability and applicability of PdM systems.

Practically, the deployment of PdM systems faces challenges related to system integration and operational scalability. It is recommended that future efforts include the development of modular systems that can be easily adapted to different technological environments. Pilot testing these systems in a variety of industrial settings would help refine their design and functionality, ensuring they meet the diverse needs of potential users. Furthermore, developing training programs for end-users on the operation and maintenance of these systems could promote smoother implementation and better results.

The complexity of PdM systems calls for interdisciplinary approaches. Forming partnerships across industries and academic fields can drive innovation and practical application. Establishing a consortium that includes data scientists, maintenance engineers, operational managers, and IT specialists could foster a collaborative approach to overcoming the multifaceted challenges of PdM. Such collaborations could also lead to the standardization of practices and enhance the transfer of knowledge across sectors.

Based on the limitations discussed, several areas require further exploration. These include the development of predictive models that can dynamically adjust to new data inputs, the exploration of AI-driven predictive analytics, and the study of the socio-economic impacts of implementing PdM. Initiating phased research projects that focus sequentially on these areas could provide systematic insights and progressive improvements in PdM technologies.

In conclusion, the path forward for PdM in industrial applications is paved with complex challenges that demand innovative solutions. It is essential for researchers, industry practitioners, and policymakers to collaboratively engage in these future initiatives. By addressing the limitations outlined in this thesis and exploring the proposed next steps, the field can move towards more reliable, efficient, and universally applicable PdM strategies. The insights and methodologies developed through these efforts will significantly contribute to the sustainability and productivity of industrial operations worldwide.

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence	1, 2, 6, 7, 13, 14, 26, 27, 28, 29, 30, 34, 42, 47, 50, 62, 88, 89, 118, 157, 162, 163, 166, 181, 182, 183, 184, 186, 194
ANN	Artificial Neural Network	32, 43, 92
AST	Abstract Syntax Tree	163
AutoML	Automated Machine Learning	93, 94
C_V	Confirmatory V-measure	36, 37, 38, 39, ix
C-Bus	Communication Bus	116, 117
CbM	Condition-based Monitoring	3
CBSE	Component-Based Software Engineering	10, 13, 14, 15, 17, 18, 19, 20, 21, 25, 26, 27, 28, 29, 30, 43, 44, 47, 54, 55, 61, 62, 63, 78, 95, 96, 108, 113, 117, 118, 119, 120, 125, 126, 127, 185, 186, 188
CD	Continuous Delivery	23
CI	Continuous Integration	23
CM	Condition Monitoring	54
CMS	Condition Monitoring System	42
CNN	Convolutional Neural Network	114, 115, 116, 118
CPPSS	Cyber-Physical Product Service System	4, 5
CPS	Cyber-Physical System	4, 163
CRISP-DM	Cross Industry Standard Process for Data Mining	89, 90, ix
CT	Continuous Training	23
DA	Data Analytics	95, 106, x
DAG	Directed Acyclic Graph	94
DC	Data Source Connection	96, 97, 104, 106
DE	Data Engineering	95, 106, 191, x
DES	Discrete Event Simulation	13, 80, 130, 131, 132, 133, 134, 135, xiii
DF	*pandas* DataFrame	31
DI	Data Input	97, 98, 99, 115, 117
Dp	Dependencies	29, 76, 80, 97, 106, 137, 171, 176
DP	Data Preprocessing	97, 98, 105, 115, 117, 172
DRL	Deep Reinforcement Learning	24
DS	Data Science	94, 95, 106, x
DSR	Design Science Research	10, 11, 12, 14, 57, 58, 59, 60, 61, 62, 63, 66, 67, 68, 69, 70, 71, 72, 73, 74, 76, 78, 80, 82, 85, 86, 114, ix
E/E	Electrical/Electronical	64
ECU	Electronic Control Unit	63, 64, 65, 88
EMI	Existance of Meta-Information	109, 110
ESP	Electronic Stability Program	88
FC	Feature Creation	99, 100, 105, 115, 117, 177, 178, 180
FE	Feature Encoding	93, 99, 101, 105
FL	Feature Labeling	100, 105, 115, 117, 178
FV	Feature Visualization	99, 105, 171
GNN	Graph Neural Network	163, 164

HVAC Heating, Ventilation and Air Conditioning 88

I4.0 Industry 4.0 .... 3, 5, 25, 42, 174, 186
IIoT Industrial Internet of Things 24, 42
IoT Internet of Things 41
IS Information Systems 59, 66, 67

LDA Latent Dirichlet Allocation 14, 19, 20, 34, 35, 36, 38, 43, 44, 45, 47, 49, 51
LoC Lines of Code .... 108, 114, 115, 116, 117, 118, xiii
Lst. Listing 144, 146, 147, 149
LSTM Long Short Term Memory Network 43

MB Model Building 93, 97, 101, 102, 104, 115, 117, 171, 173
MD Model Deployment 46, 93, 94, 95, 103, 106
ME Model Evaluation 99, 102, 104, 106, 115, 117, 118, 167, 171
MeIL Mean Identifier Length 111
MEL Melody 114, 116
MIC Mean Identifier Consistency 111
ML Machine Learning 1, 2, 4, 5, 17, 18, 20, 22, 23, 24, 25, 33, 41, 42, 47, 50, 54, 62, 88, 89, 93, 94, 97, 98, 100, 101, 102, 105, 106, 112, 114, 115, 131, 132, 162, 163, 167, 170, 174, 179, 184, 186
MLOps Machine Learning Operations 22, 23, 93, 94
MSC_A Mean String Commonality 111

NLP Natural Language Processing 14, 74, 75, 163
NPMI Normalized Pointwise Mutual Information .... 36, 37, 38, 39, ix

OaT One-at-a-Time 157
OEM Original Equipment Manufacturer 11, 12, 64, 65, 88
OOSE Object-Oriented Software Engineering 107
OTA Over-The-Air 3, 63

PAISE Process Model for AI Systems Engineering 29
PdM Predictive Maintenance 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 38, 40, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 57, 58, 60, 61, 62, 63, 64, 65, 66, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 111, 112, 113, 116, 117, 118, 119, 120, 125, 145, 156, 157, 158, 161, 162, 166, 167, 169, 170, 172, 174, 175, 176, 177, 178, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 193, 194, ix, xiii
PMI Pointwise Mutual Information 36, 37
PROMETHEE Preference Ranking Organization METHod for Enrichment Evaluations 27, 28
PSS Product-Service System 4, 26
PvM Preventive Maintenance 20

RCC Component Adaptability Rate 110
RCC_f Component Function Adaptability Rate 111
RM Reactive Maintenance 20
ROI Return on Investment 71, 93
RQ Research Question 8, 9, 10, 187, 188, 189
RUL Remaining Useful Life 50, 52, 53, 54, 105

SCCp Self-Completeness-of-Component-Parameter-Value 111
SCCr Self-Completeness-of-Component-Return-Value 111

TRL Technology Readiness Level 5, 6, 7, 22

UCI UMass Coherence Index . . . . 36, 37, 38, 39, ix
UMASS UMass Coherence . . . . 36, 37, 38, 39, ix
US Utility and Setup 103, 104

VSD Value Sensitive Design 83, 84, 85, 86

Bibliography

Abadi, Martín et al. (2016). "TensorFlow: A System for Large-Scale Machine Learning". Version 2. In: DOI: 10.48550/ARXIV.1605.08695. URL: https://arxiv.org/abs/1605.08695 (visited on 02/03/2024).

Abdelli, Khouloud, Helmut Griesser, and Stephan Pachnicke (2022). "A Machine Learning-based Framework for Predictive Maintenance of Semiconductor Laser for Optical Communication". Version 1. In: DOI: 10.48550/ARXIV.2211.02842. URL: https://arxiv.org/abs/2211.02842 (visited on 01/03/2024).

Abidi, Mustufa Haider, Muneer Khan Mohammed, and Hisham Alkhalefah (Mar. 14, 2022). "Predictive Maintenance Planning for Industry 4.0 Using Machine Learning for Sustainable Manufacturing". In: *Sustainability* 14.6, p. 3387. ISSN: 2071-1050. DOI: 10.3390/su14063387. URL: https://www.mdpi.com/2071-1050/14/6/3387 (visited on 11/01/2023).

Aboukhalil, Robert (2014). "The Rising Trend in Authorship". In: *The Winnower*. ISSN: 2373-146X. DOI: 10.15200/winn.141832.26907. URL: https://authorea.com/doi/full/10.15200/winn.141832.26907 (visited on 08/25/2023).

Achouch, Mounia et al. (Aug. 12, 2022). "On Predictive Maintenance in Industry 4.0: Overview, Models, and Challenges". In: *Applied Sciences* 12.16, p. 8081. ISSN: 2076-3417. DOI: 10.3390/app12168081. URL: https://www.mdpi.com/2076-3417/12/16/8081 (visited on 08/24/2023).

Adeel Mannan, Muhammad et al. (2019). "Data Management and Visualization Using Big Data Analytics". In: *Data Science and Digital Business*. Ed. by Fausto Pedro García Márquez and Benjamin Lev. Cham: Springer International Publishing, pp. 7–22. ISBN: 978-3-319-95650-3 978-3-319-95651-0. DOI: 10.1007/978-3-319-95651-0_1. URL: http://link.springer.com/10.1007/978-3-319-95651-0_1 (visited on 01/04/2024).

Aivaliotis, P., Z. Arkouli, K. Georgoulias, et al. (Oct. 2021). "Degradation Curves Integration in Physics-Based Models: Towards the Predictive Maintenance of Industrial Robots". In: *Robotics and Computer-Integrated Manufacturing* 71, p. 102177. ISSN: 07365845. DOI: 10.1016/j.rcim.2021.102177. URL: https://linkinghub.elsevier.com/retrieve/pii/S0736584521000600 (visited on 08/25/2023).

Aivaliotis, P., Z. Arkouli, D. Kaliakatsos-Georgopoulos, et al. (2021). "Prediction Assessment Methodology for Maintenance Applications in Manufacturing". In: *Procedia CIRP* 104, pp. 1494–1499. ISSN: 22128271. DOI: 10.1016/j.procir.2021.11.252. URL: https://linkinghub.elsevier.com/retrieve/pii/S2212827121011501 (visited on 08/25/2023).

Aivaliotis, P., K. Georgoulias, et al. (2019). "Methodology for Enabling Digital Twin Using Advanced Physics-Based Modelling in Predictive Maintenance". In: *Procedia CIRP* 81, pp. 417–422. ISSN: 22128271. DOI: 10.1016/j.procir.2019.03.072. URL: https://linkinghub.elsevier.com/retrieve/pii/S2212827119303774 (visited on 08/25/2023).

Aivaliotis, P., E. Xanthakis, and A. Sardelis (2020). "Machines' Behaviour Prediction Tool (BPT) for Maintenance Applications". In: *IFAC-PapersOnLine* 53.3, pp. 325–329. ISSN: 24058963. DOI: 10.1016/j.ifacol.2020.11.052. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405896320302020 (visited on 08/25/2023).

Alam, S (Feb. 3, 2024). *Kedro*. LF AI & Data Foundation c/o The Linux Foundation.

Ali, Moez (Feb. 3, 2024). *PyCaret*.

Allamanis, Miltiadis et al. (2017). "A Survey of Machine Learning for Big Code and Naturalness". Version 2. In: DOI: 10.48550/ARXIV.1709.06182. URL: https://arxiv.org/abs/1709.06182 (visited on 02/25/2024).

Aremu, Oluseun Omotola, Roya Allison Cody, et al. (July 2020). “A Relative Entropy Based Feature Selection Framework for Asset Data in Predictive Maintenance”. In: *Computers & Industrial Engineering* 145, p. 106536. ISSN: 03608352. DOI: 10.1016/j.cie.2020.106536. URL: https://linkinghub.elsevier.com/retrieve/pii/S0360835220302709 (visited on 08/25/2023).

Aremu, Oluseun Omotola, David Hyland-Wood, and Peter Ross McAree (Apr. 2019). “A Relative Entropy Weibull-SAX Framework for Health Indices Construction and Health Stage Division in Degradation Modeling of Multivariate Time Series Asset Data”. In: *Advanced Engineering Informatics* 40, pp. 121–134. ISSN: 14740346. DOI: 10.1016/j.aei.2019.03.003. URL: https://linkinghub.elsevier.com/retrieve/pii/S1474034618305603 (visited on 08/25/2023).

— (Mar. 2020). “A Machine Learning Approach to Circumventing the Curse of Dimensionality in Discontinuous Time Series Machine Data”. In: *Reliability Engineering & System Safety* 195, p. 106706. ISSN: 09518320. DOI: 10.1016/j.ress.2019.106706. URL: https://linkinghub.elsevier.com/retrieve/pii/S0951832019304752 (visited on 08/25/2023).

Aremu, Oluseun Omotola, Adrià Salvador Palau, et al. (2018). “Structuring Data for Intelligent Predictive Maintenance in Asset Management”. In: *IFAC-PapersOnLine* 51.11, pp. 514–519. ISSN: 24058963. DOI: 10.1016/j.ifacol.2018.08.370. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405896318314952 (visited on 08/25/2023).

Arena, Fabio et al. (Dec. 31, 2021). “Predictive Maintenance in the Automotive Sector: A Literature Review”. In: *Mathematical and Computational Applications* 27.1, p. 2. ISSN: 2297-8747. DOI: 10.3390/mca27010002. URL: https://www.mdpi.com/2297-8747/27/1/2 (visited on 11/05/2023).

Arora, Sanjeev et al. (2012). “A Practical Algorithm for Topic Modeling with Provable Guarantees”. Version 1. In: DOI: 10.48550/ARXIV.1212.4777. URL: https://arxiv.org/abs/1212.4777 (visited on 08/27/2023).

Atkinson, Colin and Oliver Hummel (June 25, 2012). “Iterative and Incremental Development of Component-Based Software Architectures”. In: *Proceedings of the 15th ACM SIGSOFT Symposium on Component Based Software Engineering.* Comparch ’12: Federated Events on Component-Based Software Engineering and Software Architecture. Bertinoro Italy: ACM, pp. 77–82. ISBN: 978-1-4503-1345-2. DOI: 10.1145/2304736.2304750. URL: https://dl.acm.org/doi/10.1145/2304736.2304750 (visited on 10/15/2023).

Atzori, Luigi, Antonio Iera, and Giacomo Morabito (Oct. 2010). “The Internet of Things: A Survey”. In: *Computer Networks* 54.15, pp. 2787–2805. ISSN: 13891286. DOI: 10.1016/j.comnet.2010.05.010. URL: https://linkinghub.elsevier.com/retrieve/pii/S1389128610001568 (visited on 08/24/2023).

“Automatic Evaluation of Topic Coherence” (2006). In: Newman, David et al. *COLING-ACL 2006: 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics; 17 - 21 July 2006, Sydney, Australia; Proceedings of the Conference. Vol. 1.* Vol. 1. Stroudsburg, Pa: Association for Computational Linguistics (ACL), pp. 100–108. ISBN: 978-1-932432-65-7.

AutoML, Cloud (Feb. 3, 2024). *Cloud AutoML.* Google LLC.

Balali, Farhad, Hamid Seifoddini, and Adel Nasiri (Mar. 2020). “Data-Driven Predictive Model of Reliability Estimation Using Degradation Models: A Review”. In: *Life Cycle Reliability and Safety Engineering* 9.1, pp. 113–125. ISSN: 2520-1352, 2520-1360. DOI: 10.1007/s41872-020-00111-6. URL: http://link.springer.com/10.1007/s41872-020-00111-6 (visited on 02/11/2021).

Baldwin, Carliss Y. and Kim B. Clark (2000). *Design Rules: The Power of Modularity.* The MIT Press. ISBN: 978-0-262-26764-9. DOI: 10.7551/mitpress/2366.001.0001. URL: https://direct.mit.edu/books/book/1856/design-rulesthe-power-of-modularity (visited on 08/24/2023).

Bandur, Victor et al. (Feb. 2021). “Making the Case for Centralized Automotive E/E Architectures”. In: *IEEE Transactions on Vehicular Technology* 70.2, pp. 1230–1245. ISSN: 0018-9545, 1939-9359. DOI: 10.1109/TVT.2021.3054934. URL: https://ieeexplore.ieee.org/document/9337216/ (visited on 11/05/2023).

Banerjee, Indradumna et al. (2023). “MLOps with Enhanced Performance Control and Observability”. Version 1. In: DOI: 10.48550/ARXIV.2302.01061. URL: https://arxiv.org/abs/2302.01061 (visited on 11/01/2023).

Barlas, Panagiotis, Ivor Lanning, and Cathal Heavey (Aug. 10, 2015). “A Survey of Open Source Data Science Tools”. In: *International Journal of Intelligent Computing and Cybernetics* 8.3. Ed. by Keshav Dahal and Professor Professor Yacine Ouzrout, pp. 232–261. ISSN: 1756-378X. DOI: 10.1108/IJICC-07-2014-0031. URL: https://www.emerald.com/insight/content/doi/10.1108/IJICC-07-2014-0031/full/html (visited on 03/02/2024).

Barley, William C, Jeffrey W Treem, and Paul M Leonardi (Mar. 20, 2020). “Experts at Coordination: Examining the Performance, Production, and Value of Process Expertise”. In: *Journal of Communication* 70.1, pp. 60–89. ISSN: 0021-9916, 1460-2466. DOI: 10.1093/joc/jqz041. URL: https://academic.oup.com/joc/article/70/1/60/5739593 (visited on 11/05/2023).

Bawa, R. K. and Iqbaldeep Kaur (Dec. 29, 2016). “Algorithmic Approach for Efficient Retrieval of Component Repositories in Component Based Software Engineering”. In: *Indian Journal of Science and Technology* 9.48. ISSN: 0974-5645, 0974-6846. DOI: 10.17485/ijst/2016/v9i48/102434. URL: https://indjst.org/articles/algorithmic-approach-for-efficient-retrieval-of-component-repositories-in-component-based-software-engineering (visited on 10/15/2023).

Bekar, Ebru Turanoglu, Per Nyqvist, and Anders Skoogh (May 2020). “An Intelligent Approach for Data Pre-Processing and Analysis in Predictive Maintenance with an Industrial Case Study”. In: *Advances in Mechanical Engineering* 12.5, p. 168781402091920. ISSN: 1687-8140, 1687-8140. DOI: 10.1177/1687814020919207. URL: http://journals.sagepub.com/doi/10.1177/1687814020919207 (visited on 11/05/2023).

Berkovich, Marina et al. (June 2014). “A Requirements Data Model for Product Service Systems”. In: *Requirements Engineering* 19.2, pp. 161–186. ISSN: 0947-3602, 1432-010X. DOI: 10.1007/s00766-012-0164-1. URL: http://link.springer.com/10.1007/s00766-012-0164-1 (visited on 10/15/2023).

Bertolino, Antonia et al. (July 2008). “Use Case Description of Requirements for Product Lines”. In:

Biewald, Lukas (Sept. 18, 2023). *Machine Learning Experiment Tracking*. W&B Fully Connected. URL: https://wandb.ai/wandb_fc/articles/reports/Machine-Learning-Experiment-Tracking--Vmlldzo1NDI1Mjcy (visited on 02/03/2024).

Biggerstaff, Ted J. and Alan J. Perlis, eds. (1989). *Software Reusability*. ACM Press Frontier Series. New York, N.Y. : Reading, Mass: ACM Press ; Addison-Wesley. 2 pp. ISBN: 978-0-201-08017-9 978-0-201-50018-9.

Bin Hu et al. (July 2012). “A Two-Stage Equipment Predictive Maintenance Framework for High-Performance Manufacturing Systems”. In: *2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. 2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA). Singapore, Singapore: IEEE, pp. 1343–1348. ISBN: 978-1-4577-2119-9 978-1-4577-2118-2 978-1-4577-2117-5. DOI: 10.1109/ICIEA.2012.6360931. URL: http://ieeexplore.ieee.org/document/6360931/ (visited on 11/25/2023).

Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. Information Science and Statistics. New York: Springer. ISBN: 978-0-387-31073-2.

Blair, Stuart J., Yaxin Bi, and Maurice D. Mulvenna (Jan. 2020). “Aggregated Topic Models for Increasing Social Media Topic Coherence”. In: *Applied Intelligence* 50.1, pp. 138–156. ISSN:

0924-669X, 1573-7497. DOI: 10.1007/s10489-019-01438-z. URL: http://link.springer.com/10.1007/s10489-019-01438-z (visited on 08/27/2023).

Blei, David M., Andrew Y. Ng, and Michael I. Jordan (2003). “Latent Dirichlet Allocation”. In: *Journal of Machine Learning Research* 3, pp. 993–1022. URL: https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf (visited on 08/27/2023).

Boardman, J. and B. Sauser (2006). “System of Systems - the Meaning of Of”. In: *2006 IEEE/SMC International Conference on System of Systems Engineering.* 2006 IEEE/SMC International Conference on System of Systems Engineering. Los Angeles, California, USA: IEEE, pp. 118–123. ISBN: 978-1-4244-0188-8. DOI: 10.1109/SYSOSE.2006.1652284. URL: http://ieeexplore.ieee.org/document/1652284/ (visited on 08/24/2023).

Booth, Wayne C. et al. (2016). *The Craft of Research.* Fourth edition. Chicago Guides to Writing, Editing, and Publishing. Chicago: The University of Chicago Press. 316 pp. ISBN: 978-0-226-23956-9 978-0-226-23973-6.

Bortolotti, Roberta (2018). “Data Prep 1–2: Data Description”. In: *Handbook of Statistical Analysis and Data Mining Applications.* Elsevier, pp. 459–469. ISBN: 978-0-12-416632-5. DOI: 10.1016/B978-0-12-416632-5.00031-1. URL: https://linkinghub.elsevier.com/retrieve/pii/B9780124166325000311 (visited on 03/02/2024).

Bosch, Jan (2000). *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach.* 4. [print.] ACM Press Books. Harlow: Addison-Wesley. 354 pp. ISBN: 978-0-201-67494-1.

Bouabdallaoui, Yassine et al. (Feb. 3, 2021). “Predictive Maintenance in Building Facilities: A Machine Learning-Based Approach”. In: *Sensors* 21.4, p. 1044. ISSN: 1424-8220. DOI: 10.3390/s21041044. URL: https://www.mdpi.com/1424-8220/21/4/1044 (visited on 11/01/2023).

Bousdekis, Alexandros, Dimitris Apostolou, and Gregoris Mentzas (Mar. 1, 2020). “Predictive Maintenance in the 4th Industrial Revolution: Benefits, Business Opportunities, and Managerial Implications”. In: *IEEE Engineering Management Review* 48.1, pp. 57–62. ISSN: 0360-8581, 1937-4178. DOI: 10.1109/EMR.2019.2958037. URL: https://ieeexplore.ieee.org/document/8930914/ (visited on 08/24/2023).

Boxall, M.A.S. and S. Araban (2004). “Interface Metrics for Reusability Analysis of Components”. In: *2004 Australian Software Engineering Conference. Proceedings.* 2004 Australian Software Engineering Conference. Proceedings. Melbourne, Vic., Australia: IEEE, pp. 40–51. ISBN: 978-0-7695-2089-6. DOI: 10.1109/ASWEC.2004.1290456. URL: http://ieeexplore.ieee.org/document/1290456/ (visited on 01/16/2024).

Bunks, Carey, Dan Mccarthy, and Tarik Al-Ani (July 2000). “CONDITION-BASED MAINTENANCE OF MACHINES USING HIDDEN MARKOV MODELS”. In: *Mechanical Systems and Signal Processing* 14.4, pp. 597–612. ISSN: 08883270. DOI: 10.1006/mssp.2000.1309. URL: https://linkinghub.elsevier.com/retrieve/pii/S0888327000913098 (visited on 11/25/2023).

Byrum, Joseph et al. (Feb. 2016). “Advanced Analytics for Agricultural Product Development”. In: *Interfaces* 46.1, pp. 5–17. ISSN: 0092-2102, 1526-551X. DOI: 10.1287/inte.2015.0823. URL: https://pubsonline.informs.org/doi/10.1287/inte.2015.0823 (visited on 02/24/2024).

Cadavid, Usuga et al. (June 3, 2022). “Valuing Free-Form Text Data from Maintenance Logs through Transfer Learning with CamemBERT”. In: *Enterprise Information Systems* 16.6, p. 1790043. ISSN: 1751-7575, 1751-7583. DOI: 10.1080/17517575.2020.1790043. URL: https://www.tandfonline.com/doi/full/10.1080/17517575.2020.1790043 (visited on 11/01/2023).

Cai, Xia, Michael R. Lyu, and Kam-Fai Wong (Apr. 2002). “COMPONENT-BASED EMBEDDED SOFTWARE ENGINEERING: DEVELOPMENT FRAMEWORK, QUALITY ASSURANCE AND A GENERIC ASSESSMENT ENVIRONMENT”. In: *International*

Journal of Software Engineering and Knowledge Engineering 12.02, pp. 107–133. ISSN: 0218-1940, 1793-6403. DOI: `10.1142/S0218194002000846`. URL: `https://www.worldscientific.com/doi/abs/10.1142/S0218194002000846` (visited on 10/15/2023).

Cakir, Mustafa, Mehmet Ali Guvenc, and Selcuk Mistikoglu (Jan. 2021). "The Experimental Application of Popular Machine Learning Algorithms on Predictive Maintenance and the Design of IIoT Based Condition Monitoring System". In: *Computers & Industrial Engineering* 151, p. 106948. ISSN: 03608352. DOI: `10.1016/j.cie.2020.106948`. URL: `https://linkinghub.elsevier.com/retrieve/pii/S0360835220306252` (visited on 10/27/2023).

Cao, Juan et al. (Mar. 2009). "A Density-Based Method for Adaptive LDA Model Selection". In: *Neurocomputing* 72.7-9, pp. 1775–1781. ISSN: 09252312. DOI: `10.1016/j.neucom.2008.06.011`. URL: `https://linkinghub.elsevier.com/retrieve/pii/S092523120800372X` (visited on 10/28/2023).

Carvalho, Thyago P. et al. (Nov. 2019). "A Systematic Literature Review of Machine Learning Methods Applied to Predictive Maintenance". In: *Computers & Industrial Engineering* 137, p. 106024. ISSN: 03608352. DOI: `10.1016/j.cie.2019.106024`. URL: `https://linkinghub.elsevier.com/retrieve/pii/S0360835219304838` (visited on 02/11/2021).

Castro, João Aguiar et al. (2015). "Ontologies for Research Data Description: A Design Process Applied to Vehicle Simulation". In: *Metadata and Semantics Research*. Ed. by Emmanouel Garoufallou, Richard J. Hartley, and Panorea Gaitanou. Vol. 544. Cham: Springer International Publishing, pp. 348–354. ISBN: 978-3-319-24128-9 978-3-319-24129-6. DOI: `10.1007/978-3-319-24129-6_30`. URL: `http://link.springer.com/10.1007/978-3-319-24129-6_30` (visited on 03/02/2024).

Çetinkaya, Deniz, Alexander Verbraeck, and Mamadou D. Seck (May 7, 2015). "Model Continuity in Discrete Event Simulation: A Framework for Model-Driven Development of Simulation Models". In: *ACM Transactions on Modeling and Computer Simulation* 25.3, pp. 1–24. ISSN: 1049-3301, 1558-1195. DOI: `10.1145/2699714`. URL: `https://dl.acm.org/doi/10.1145/2699714` (visited on 02/24/2024).

Chande, P.K. and S.V. Tokekar (Mar. 1998). "Expert-Based Maintenance: A Study of Its Effectiveness". In: *IEEE Transactions on Reliability* 47.1, pp. 53–58. ISSN: 00189529. DOI: `10.1109/24.690904`. URL: `http://ieeexplore.ieee.org/document/690904/` (visited on 11/05/2023).

Chang, Jonathan et al. (2009). "Reading Tea Leaves: How Humans Interpret Topic Models". In: *Proceedings of the 22nd International Conference on Neural Information Processing Systems*. NIPS'09. Red Hook, NY, USA: Curran Associates Inc., pp. 288–296. ISBN: 978-1-61567-911-9.

Chapman, Andrea L., Laura C. Morgan, and Gerald Gartlehner (Mar. 2010). "Semi-Automating the Manual Literature Search for Systematic Reviews Increases Efficiency". In: *Health Information & Libraries Journal* 27.1, pp. 22–27. ISSN: 14711834, 14711842. DOI: `10.1111/j.1471-1842.2009.00865.x`. URL: `https://onlinelibrary.wiley.com/doi/10.1111/j.1471-1842.2009.00865.x` (visited on 08/25/2023).

Che, Yunhong et al. (Feb. 2021). "Predictive Battery Health Management With Transfer Learning and Online Model Correction". In: *IEEE Transactions on Vehicular Technology* 70.2, pp. 1269–1277. ISSN: 0018-9545, 1939-9359. DOI: `10.1109/TVT.2021.3055811`. URL: `https://ieeexplore.ieee.org/document/9343713/` (visited on 01/02/2024).

Chen, Chuang et al. (June 30, 2021). "A Data-Driven Predictive Maintenance Strategy Based on Accurate Failure Prognostics". In: *Eksploatacja i Niezawodność – Maintenance and Reliability* 23.2, pp. 387–394. ISSN: 1507-2711, 2956-3860. DOI: `10.17531/ein.2021.2.19`. URL: `https://ein.org.pl/A-data-driven-predictive-maintenance-strategy-based-on-accurate-failure-prognostics,158343,0,2.html` (visited on 08/24/2023).

Chen, Chun-Hsien et al., eds. (2017). *Transdisciplinary Engineering: A Paradigm Shift : Proceedings of the 24th ISPE Inc. International Conference on Transdisciplinary Engineering, July 10-14, 2017*. Amsterdam: IOS Press. ISBN: 978-1-61499-779-5.

Chen, WenShin and Rudy Hirschheim (July 2004). "A Paradigmatic and Methodological Examination of Information Systems Research from 1991 to 2001". In: *Information Systems Journal* 14.3, pp. 197–235. ISSN: 1350-1917, 1365-2575. DOI: `10.1111/j.1365-2575.2004.00173.x`. URL: `https://onlinelibrary.wiley.com/doi/10.1111/j.1365-2575.2004.00173.x` (visited on 04/14/2024).

Cheng, Jack C.P. et al. (Apr. 2020). "Data-Driven Predictive Maintenance Planning Framework for MEP Components Based on BIM and IoT Using Machine Learning Algorithms". In: *Automation in Construction* 112, p. 103087. ISSN: 09265805. DOI: `10.1016/j.autcon.2020.103087`. URL: `https://linkinghub.elsevier.com/retrieve/pii/S0926580518308562` (visited on 10/28/2023).

Chilana, Parmit K., Jacob O. Wobbrock, and Amy J. Ko (Apr. 10, 2010). "Understanding Usability Practices in Complex Domains". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '10: CHI Conference on Human Factors in Computing Systems. Atlanta Georgia USA: ACM, pp. 2337–2346. ISBN: 978-1-60558-929-9. DOI: `10.1145/1753326.1753678`. URL: `https://dl.acm.org/doi/10.1145/1753326.1753678` (visited on 11/05/2023).

Chuang, Shang-Yi et al. (Sept. 9, 2019). "Predictive Maintenance with Sensor Data Analytics on a Raspberry Pi-Based Experimental Platform". In: *Sensors* 19.18, p. 3884. ISSN: 1424-8220. DOI: `10.3390/s19183884`. URL: `https://www.mdpi.com/1424-8220/19/18/3884` (visited on 10/28/2023).

Çınar, Zeki Murat et al. (Oct. 5, 2020). "Machine Learning in Predictive Maintenance towards Sustainable Smart Manufacturing in Industry 4.0". In: *Sustainability* 12.19, p. 8211. ISSN: 2071-1050. DOI: `10.3390/su12198211`. URL: `https://www.mdpi.com/2071-1050/12/19/8211` (visited on 11/19/2023).

Clements, Paul and Linda Northrop (2002). *Software Product Lines: Practices and Patterns*. The SEI Series in Software Engineering. Boston: Addison-Wesley. 563 pp. ISBN: 978-0-201-70332-0.

Cleven, Anne, Philipp Gubler, and Kai M. Hüner (2009). "Design Alternatives for the Evaluation of Design Science Research Artifacts". In: *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology - DESRIST '09*. The 4th International Conference. Philadelphia, Pennsylvania: ACM Press, p. 1. ISBN: 978-1-60558-408-9. DOI: `10.1145/1555619.1555645`. URL: `http://portal.acm.org/citation.cfm?doid=1555619.1555645` (visited on 02/11/2021).

Cockburn, Alistair (2012). *Writing Effective Use Cases*. 24. print. The Agile Software Development Series. Boston: Addison-Wesley. 270 pp. ISBN: 978-0-201-70225-5.

Cohen, Sholom and Robert Krut (2018). "Managing Variation in Services in a Software Product Line Context". In: 570990 Bytes. DOI: `10.1184/R1/6575261.V1`. URL: `https://figshare.com/articles/Managing_Variation_in_Services_in_a_Software_Product_Line_Context/6575261/1` (visited on 08/24/2023).

Conmy, Philippa and Iain Bate (Jan. 2014). "Assuring Safety for Component Based Software Engineering". In: *2014 IEEE 15th International Symposium on High-Assurance Systems Engineering*. 2014 IEEE 15th International Symposium on High-Assurance Systems Engineering (HASE). Miami Beach, FL, USA: IEEE, pp. 121–128. ISBN: 978-1-4799-3466-9 978-1-4799-3465-2. DOI: `10.1109/HASE.2014.25`. URL: `http://ieeexplore.ieee.org/document/6754596/` (visited on 10/15/2023).

Contreras-Piña, Constanza and Sebastián A. Ríos (Feb. 2016). "An Empirical Comparison of Latent Sematic Models for Applications in Industry". In: *Neurocomputing* 179, pp. 176–185. ISSN: 09252312. DOI: `10.1016/j.neucom.2015.11.080`. URL: `https://linkinghub.elsevier.com/retrieve/pii/S0925231215019062` (visited on 08/27/2023).

Creswell, John W. (2014). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. 4th ed. Thousand Oaks: SAGE Publications. 273 pp. ISBN: 978-1-4522-2609-5 978-1-4522-2610-1.

Crnkovic, Ivica and Magnus Larsson (2003). *Building Reliable Component-Based Software Systems*. Norwood: Artech House. ISBN: 978-1-58053-558-8.

Crnkovic, Ivica, Severine Sentilles, et al. (Sept. 2011). "A Classification Framework for Software Component Models". In: *IEEE Transactions on Software Engineering* 37.5, pp. 593–615. ISSN: 0098-5589, 1939-3520. DOI: 10.1109/TSE.2010.83. URL: http://ieeexplore.ieee.org/document/5587419/ (visited on 10/15/2023).

Crnkovic, Ivica, Judith A Stafford, et al. (2004). *Component-Based Software Engineering: 7th International Symposium, CBSE 2004, Edinburgh, UK, May 24-25, 2004. Proceedings*. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg : Springer e-books. ISBN: 978-3-540-24774-6.

Dalzochio, Jovani et al. (Dec. 2020). "Machine Learning and Reasoning for Predictive Maintenance in Industry 4.0: Current Status and Challenges". In: *Computers in Industry* 123, p. 103298. ISSN: 01663615. DOI: 10.1016/j.compind.2020.103298. URL: https://linkinghub.elsevier.com/retrieve/pii/S0166361520305327 (visited on 08/24/2023).

Dasu, Tamraparni and Theodore Johnson (May 9, 2003). *Exploratory Data Mining and Data Cleaning*. 1st ed. Wiley Series in Probability and Statistics. Wiley. ISBN: 978-0-471-26851-2 978-0-471-44835-8. DOI: 10.1002/0471448354. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/0471448354 (visited on 11/25/2023).

Davari, Narjes et al. (Aug. 26, 2021). "A Survey on Data-Driven Predictive Maintenance for the Railway Industry". In: *Sensors* 21.17, p. 5739. ISSN: 1424-8220. DOI: 10.3390/s21175739. URL: https://www.mdpi.com/1424-8220/21/17/5739 (visited on 03/09/2024).

De Santo, Aniello et al. (Dec. 2022). "Evaluating Time Series Encoding Techniques for Predictive Maintenance". In: *Expert Systems with Applications* 210, p. 118435. ISSN: 09574174. DOI: 10.1016/j.eswa.2022.118435. URL: https://linkinghub.elsevier.com/retrieve/pii/S0957417422015342 (visited on 01/02/2024).

De Souza, Alessandro J. and Anderson Luiz O. Cavalcanti (Sept. 2016). "Visual Language for Use Case Description: VISUAL LANGUAGE FOR USE CASE DESCRIPTION". In: *Software: Practice and Experience* 46.9, pp. 1239–1261. ISSN: 00380644. DOI: 10.1002/spe.2376. URL: https://onlinelibrary.wiley.com/doi/10.1002/spe.2376 (visited on 03/30/2024).

Decker, Leticia et al. (May 2020). "Big Data Analysis for Predictive Maintenance at the INFN-CNAF Data Center Using Machine Learning Approaches". In: *PROCEEDING OF THE 25TH CONFERENCE OF FRUCT ASSOCIATION*, pp. 448–451. ISSN: 2305-7254.

Demetriadis, Stavros et al. (Mar. 2011). "Peer Review-Based Scripted Collaboration to Support Domain-Specific and Domain-General Knowledge Acquisition in Computer Science". In: *Computer Science Education* 21.1, pp. 29–56. ISSN: 0899-3408, 1744-5175. DOI: 10.1080/08993408.2010.539069. URL: http://www.tandfonline.com/doi/abs/10.1080/08993408.2010.539069 (visited on 11/05/2023).

Ding, Weicong, Prakash Ishwar, and Venkatesh Saligrama (2014). "A Topic Modeling Approach to Ranking". Version 3. In: DOI: 10.48550/ARXIV.1412.3705. URL: https://arxiv.org/abs/1412.3705 (visited on 08/27/2023).

Doogan, Caitlin and Wray Buntine (2021). "Topic Model or Topic Twaddle? Re-evaluating Semantic Interpretability Measures". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Online: Association for Computational Linguistics, pp. 3824–3848. DOI: 10.18653/v1/2021.naacl-main.300. URL: https://aclanthology.org/2021.naacl-main.300 (visited on 08/27/2023).

Doshi-Velez, Finale and Been Kim (2017). "Towards A Rigorous Science of Interpretable Machine Learning". Version 2. In: DOI: 10.48550/ARXIV.1702.08608. URL: https://arxiv.org/abs/1702.08608 (visited on 04/13/2024).

Drocco, Maurizio et al. (May 3, 2017). *A Formal Semantics for Data Analytics Pipelines.* arXiv: 1705.01629 [cs]. URL: http://arxiv.org/abs/1705.01629 (visited on 03/02/2024). preprint.

Easterbrook, Steve et al. (2008). "Selecting Empirical Methods for Software Engineering Research". In: *Guide to Advanced Empirical Software Engineering.* Ed. by Forrest Shull, Janice Singer, and Dag I. K. Sjøberg. London: Springer London, pp. 285–311. ISBN: 978-1-84800-043-8 978-1-84800-044-5. DOI: 10.1007/978-1-84800-044-5_11. URL: http://link.springer.com/10.1007/978-1-84800-044-5_11 (visited on 10/15/2023).

Echchakoui, Saïd (Sept. 2020). "Why and How to Merge Scopus and Web of Science during Bibliometric Analysis: The Case of Sales Force Literature from 1912 to 2019". In: *Journal of Marketing Analytics* 8.3, pp. 165–184. ISSN: 2050-3318, 2050-3326. DOI: 10.1057/s41270-020-00081-9. URL: https://link.springer.com/10.1057/s41270-020-00081-9 (visited on 08/25/2023).

Edmunds, Andrew, Colin Snook, and Marina Walden (2016). "On Component-Based Reuse for Event-B". In: *Abstract State Machines, Alloy, B, TLA, VDM, and Z.* Ed. by Michael Butler et al. Vol. 9675. Cham: Springer International Publishing, pp. 151–166. ISBN: 978-3-319-33599-5 978-3-319-33600-8. DOI: 10.1007/978-3-319-33600-8_9. URL: http://link.springer.com/10.1007/978-3-319-33600-8_9 (visited on 01/19/2024).

Eke, Samuel et al. (Aug. 2017). "Characterization of the Operating Periods of a Power Transformer by Clustering the Dissolved Gas Data". In: *2017 IEEE 11th International Symposium on Diagnostics for Electrical Machines, Power Electronics and Drives (SDEMPED).* 2017 IEEE 11th International Symposium on Diagnostics for Electrical Machines, Power Electronics and Drives (SDEMPED). Tinos, Greece: IEEE, pp. 298–303. ISBN: 978-1-5090-0409-6. DOI: 10.1109/DEMPED.2017.8062371. URL: http://ieeexplore.ieee.org/document/8062371/ (visited on 11/25/2023).

El Mahdi, Bouyahrouzi et al. (Dec. 31, 2022). "Real Time Assessment of Novel Predictive Maintenance System Based on Artificial Intelligence for Rotating Machines". In: *Journal Européen des Systèmes Automatisés* 55.6, pp. 817–823. ISSN: 12696935, 21167087. DOI: 10.18280/jesa.550614. URL: https://www.iieta.org/journals/jesa/paper/10.18280/jesa.550614 (visited on 10/15/2023).

Emmerich, Wolfgang and Nima Kaveh (2002). "Component Technologies: Java Beans, COM, CORBA, RMI, EJB and the CORBA Component Model". In: *Proceedings of the 24th International Conference on Software Engineering - ICSE '02.* The 24th International Conference. Orlando, Florida: ACM Press, p. 691. ISBN: 978-1-58113-472-8. DOI: 10.1145/581339.581448. URL: http://portal.acm.org/citation.cfm?doid=581339.581448 (visited on 10/15/2023).

Ewald, Vincentius et al. (Feb. 2022). "Perception Modelling by Invariant Representation of Deep Learning for Automated Structural Diagnostic in Aircraft Maintenance: A Study Case Using DeepSHM". In: *Mechanical Systems and Signal Processing* 165, p. 108153. ISSN: 08883270. DOI: 10.1016/j.ymssp.2021.108153. URL: https://linkinghub.elsevier.com/retrieve/pii/S0888327021005331 (visited on 10/27/2023).

Fakhimi, Masoud et al. (Dec. 2014). "A Hybrid Agent-Based and Discrete Event Simulation Approach for Sustainable Strategic Planning and Simulation Analytics". In: *Proceedings of the Winter Simulation Conference 2014.* 2014 Winter Simulation Conference - (WSC 2014). Savanah, GA, USA: IEEE, pp. 1573–1584. ISBN: 978-1-4799-7486-3 978-1-4799-7484-9. DOI: 10.1109/WSC.2014.7020009. URL: http://ieeexplore.ieee.org/document/7020009/ (visited on 02/24/2024).

Farrar, Charles R. and Keith Worden (Nov. 2012). *Structural Health Monitoring: A Machine Learning Perspective.* 1st ed. Wiley. ISBN: 978-1-119-99433-6 978-1-118-44311-8. DOI: 10.1002/9781118443118. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/9781118443118 (visited on 11/25/2023).

Fausing Olesen, Jonas and Hamid Reza Shaker (Apr. 24, 2020). “Predictive Maintenance for Pump Systems and Thermal Power Plants: State-of-the-Art Review, Trends and Challenges”. In: *Sensors* 20.8, p. 2425. ISSN: 1424-8220. DOI: 10.3390/s20082425. URL: https://www.mdpi.com/1424-8220/20/8/2425 (visited on 01/03/2024).

Feng, Ke, J.C. Ji, Yifan Li, et al. (July 2022). “A Novel Cyclic-Correntropy Based Indicator for Gear Wear Monitoring”. In: *Tribology International* 171, p. 107528. ISSN: 0301679X. DOI: 10.1016/j.triboint.2022.107528. URL: https://linkinghub.elsevier.com/retrieve/pii/S0301679X22001013 (visited on 08/25/2023).

Feng, Ke, J.C. Ji, Qing Ni, et al. (Jan. 2023). “A Review of Vibration-Based Gear Wear Monitoring and Prediction Techniques”. In: *Mechanical Systems and Signal Processing* 182, p. 109605. ISSN: 08883270. DOI: 10.1016/j.ymssp.2022.109605. URL: https://linkinghub.elsevier.com/retrieve/pii/S0888327022006951 (visited on 08/25/2023).

Feng, Ke, J.C. Ji, Yongchao Zhang, et al. (Mar. 2023). “Digital Twin-Driven Intelligent Assessment of Gear Surface Degradation”. In: *Mechanical Systems and Signal Processing* 186, p. 109896. ISSN: 08883270. DOI: 10.1016/j.ymssp.2022.109896. URL: https://linkinghub.elsevier.com/retrieve/pii/S0888327022009645 (visited on 08/25/2023).

Feng, Ke, Qing Ni, et al. (Oct. 2022). “A Novel Similarity-Based Status Characterization Methodology for Gear Surface Wear Propagation Monitoring”. In: *Tribology International* 174, p. 107765. ISSN: 0301679X. DOI: 10.1016/j.triboint.2022.107765. URL: https://linkinghub.elsevier.com/retrieve/pii/S0301679X22003383 (visited on 08/25/2023).

Fernandes, João M. (Oct. 2014). “Authorship Trends in Software Engineering”. In: *Scientometrics* 101.1, pp. 257–271. ISSN: 0138-9130, 1588-2861. DOI: 10.1007/s11192-014-1331-6. URL: http://link.springer.com/10.1007/s11192-014-1331-6 (visited on 08/25/2023).

Fernandes, João M. and Miguel P. Monteiro (Feb. 2017). “Evolution in the Number of Authors of Computer Science Publications”. In: *Scientometrics* 110.2, pp. 529–539. ISSN: 0138-9130, 1588-2861. DOI: 10.1007/s11192-016-2214-9. URL: http://link.springer.com/10.1007/s11192-016-2214-9 (visited on 08/25/2023).

Fernandes, Patrick, Miltiadis Allamanis, and Marc Brockschmidt (2018). “Structured Neural Summarization”. Version 4. In: DOI: 10.48550/ARXIV.1811.01824. URL: https://arxiv.org/abs/1811.01824 (visited on 02/25/2024).

Ferraro, Antonino et al. (Dec. 2020). “A Novel Approach for Predictive Maintenance Combining GAF Encoding Strategies and Deep Networks”. In: *2020 IEEE 6th International Conference on Dependability in Sensor, Cloud and Big Data Systems and Application (DependSys)*. 2020 IEEE 6th International Conference on Dependability in Sensor, Cloud and Big Data Systems and Application (DependSys). Nadi, Fiji: IEEE, pp. 127–132. ISBN: 978-1-72817-651-2. DOI: 10.1109/DependSys51298.2020.00027. URL: https://ieeexplore.ieee.org/document/9356422/ (visited on 01/02/2024).

Feurer, Matthias et al. (2020). “Auto-Sklearn 2.0: Hands-free AutoML via Meta-Learning”. Version 3. In: DOI: 10.48550/ARXIV.2007.04074. URL: https://arxiv.org/abs/2007.04074 (visited on 02/03/2024).

Frakes, William and Carol Terry (June 1996). “Software Reuse: Metrics and Models”. In: *ACM Computing Surveys* 28.2, pp. 415–435. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/234528.234531. URL: https://dl.acm.org/doi/10.1145/234528.234531 (visited on 08/24/2023).

Freitas, Leo and John McDermott (Oct. 2011). “Formal Methods for Security in the Xenon Hypervisor”. In: *International Journal on Software Tools for Technology Transfer* 13.5, pp. 463–489. ISSN: 1433-2779, 1433-2787. DOI: 10.1007/s10009-011-0195-9. URL: http://link.springer.com/10.1007/s10009-011-0195-9 (visited on 03/02/2024).

Friedman, Batya and David Hendry (2019). *Value Sensitive Design: Shaping Technology with Moral Imagination*. Cambridge, Massachusetts: The MIT Press. 229 pp. ISBN: 978-0-262-03953-6.

Fromm, Hansjörg (2020). “Industrial Services: Condition Monitoring, Predictive Maintenance”. Lecture (Karlsruhe Institute of Technology (KIT), Karlsruhe).

Fugate, Michael L., Hoon Sohn, and Charles R. Farrar (July 2001). “VIBRATION-BASED DAMAGE DETECTION USING STATISTICAL PROCESS CONTROL”. In: *Mechanical Systems and Signal Processing* 15.4, pp. 707–721. ISSN: 08883270. DOI: 10.1006/mssp.2000.1323. URL: https://linkinghub.elsevier.com/retrieve/pii/S0888327000913232 (visited on 11/25/2023).

Gan, Jingxian and Yong Qi (Oct. 3, 2021). “Selection of the Optimal Number of Topics for LDA Topic Model—Taking Patent Policy Analysis as an Example”. In: *Entropy* 23.10, p. 1301. ISSN: 1099-4300. DOI: 10.3390/e23101301. URL: https://www.mdpi.com/1099-4300/23/10/1301 (visited on 10/28/2023).

García Márquez, Fausto Pedro et al. (Oct. 2012). “Condition Monitoring of Wind Turbines: Techniques and Methods”. In: *Renewable Energy* 46, pp. 169–178. ISSN: 09601481. DOI: 10.1016/j.renene.2012.03.003. URL: https://linkinghub.elsevier.com/retrieve/pii/S0960148112001899 (visited on 11/25/2023).

Garrido Martínez-Llop, Pablo, Juan De Dios Sanz Bobi, and Manuel Olmedo Ortega (Aug. 2023). “Time Consideration in Machine Learning Models for Train Comfort Prediction Using LSTM Networks”. In: *Engineering Applications of Artificial Intelligence* 123, p. 106303. ISSN: 09521976. DOI: 10.1016/j.engappai.2023.106303. URL: https://linkinghub.elsevier.com/retrieve/pii/S0952197623004876 (visited on 10/27/2023).

Gašperin, Matej et al. (Feb. 2011). “Model-Based Prognostics of Gear Health Using Stochastic Dynamical Models”. In: *Mechanical Systems and Signal Processing* 25.2, pp. 537–548. ISSN: 08883270. DOI: 10.1016/j.ymssp.2010.07.003. URL: https://linkinghub.elsevier.com/retrieve/pii/S0888327010002396 (visited on 11/25/2023).

Gebraeel, N. (Oct. 2006). “Sensory-Updated Residual Life Distributions for Components With Exponential Degradation Patterns”. In: *IEEE Transactions on Automation Science and Engineering* 3.4, pp. 382–393. ISSN: 1545-5955, 1558-3783. DOI: 10.1109/TASE.2006.876609. URL: http://ieeexplore.ieee.org/document/1707956/ (visited on 11/25/2023).

Gebraeel, N., A. Elwany, and Jing Pan (Mar. 2009). “Residual Life Predictions in the Absence of Prior Degradation Knowledge”. In: *IEEE Transactions on Reliability* 58.1, pp. 106–117. ISSN: 0018-9529, 1558-1721. DOI: 10.1109/TR.2008.2011659. URL: http://ieeexplore.ieee.org/document/4781602/ (visited on 11/25/2023).

Gerlach, Martin, Tiago P. Peixoto, and Eduardo G. Altmann (July 6, 2018). “A Network Approach to Topic Models”. In: *Science Advances* 4.7, eaaq1360. ISSN: 2375-2548. DOI: 10.1126/sciadv.aaq1360. URL: https://www.science.org/doi/10.1126/sciadv.aaq1360 (visited on 08/27/2023).

Goode, K B, J Moore, and B J Roylance (May 1, 2000). “Plant Machinery Working Life Prediction Method Utilizing Reliability and Condition-Monitoring Data”. In: *Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering* 214.2, pp. 109–122. ISSN: 0954-4089, 2041-3009. DOI: 10.1243/0954408001530146. URL: http://journals.sagepub.com/doi/10.1243/0954408001530146 (visited on 11/25/2023).

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. Adaptive Computation and Machine Learning. Cambridge, Massachusetts London, England: The MIT Press. 775 pp. ISBN: 978-0-262-03561-3.

Gorishti, Albana and Klaidi Gorishti (2022). “A Predictive Maintenance Deployment Model for IoT Scenarios”. In: Eighth International Scientific-Business Conference LIMEN Leadership, Innovation, Management and Economics: Integrated Politics of Research, pp. 129–135. DOI: 10.31410/LIMEN.2022.129. URL: https://limen-conference.com/maintenance-deployment-model/ (visited on 01/02/2024).

Grambau, Jens, Arno Hitzges, and Boris Otto (June 2019). “Reference Architecture Framework for Enhanced Social Media Data Analytics for Predictive Maintenance Models”. In: *2019 IEEE*

International Conference on Engineering, Technology and Innovation (ICE/ITMC). 2019 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC). Valbonne Sophia-Antipolis, France: IEEE, pp. 1–8. ISBN: 978-1-72813-401-7. DOI: 10.1109/ICE.2019.8792678. URL: https://ieeexplore.ieee.org/document/8792678/ (visited on 01/25/2021).

Greasley, Andrew and John Steven Edwards (Feb. 1, 2021). “Enhancing Discrete-Event Simulation with Big Data Analytics: A Review”. In: *Journal of the Operational Research Society* 72.2, pp. 247–267. ISSN: 0160-5682, 1476-9360. DOI: 10.1080/01605682.2019.1678406. URL: https://www.tandfonline.com/doi/full/10.1080/01605682.2019.1678406 (visited on 02/24/2024).

Gregor, Shirley and Alan R. Hevner (Feb. 2, 2013). “Positioning and Presenting Design Science Research for Maximum Impact”. In: *MIS Quarterly* 37.2, pp. 337–355. ISSN: 02767783, 21629730. DOI: 10.25300/MISQ/2013/37.2.01. URL: https://misq.org/positioning-and-presenting-design-science-research-for-maximum-impact.html (visited on 10/15/2023).

Griffiths, Thomas L. and Mark Steyvers (Apr. 6, 2004). “Finding Scientific Topics”. In: *Proceedings of the National Academy of Sciences* 101 (suppl_1), pp. 5228–5235. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0307752101. URL: https://pnas.org/doi/full/10.1073/pnas.0307752101 (visited on 11/25/2023).

Groba, Christin et al. (June 2007). “Architecture of a Predictive Maintenance Framework”. In: *6th International Conference on Computer Information Systems and Industrial Management Applications (CISIM’07)*. 6th International Conference on Computer Information Systems and Industrial Management Applications (CISIM’07). Elk, Poland: IEEE, pp. 59–64. ISBN: 978-0-7695-2894-6. DOI: 10.1109/CISIM.2007.14. URL: http://ieeexplore.ieee.org/document/4273496/ (visited on 11/05/2023).

Grunske, Lars, Bernhard Kaiser, and Ralf H. Reussner (2005). “Specification and Evaluation of Safety Properties in a Component-Based Software Engineering Process”. In: *Component-Based Software Development for Embedded Systems*. Ed. by Colin Atkinson et al. Red. by David Hutchison et al. Vol. 3778. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 249–274. ISBN: 978-3-540-30644-3 978-3-540-31614-5. DOI: 10.1007/11591962_13. URL: http://link.springer.com/10.1007/11591962_13 (visited on 10/15/2023).

Gulati, Ramesh (2012). *Maintenance and Reliability Best Practices*. 2. ed. New York, NY: Industrial Press. ISBN: 978-0-8311-3434-1.

Gunes, Volkan (Dec. 31, 2014). “A Survey on Concepts, Applications, and Challenges in Cyber-Physical Systems”. In: *KSII Transactions on Internet and Information Systems* 8.12. ISSN: 19767277. DOI: 10.3837/tiis.2014.12.001. URL: http://www.itiis.org/digital-library/manuscript/894 (visited on 03/30/2024).

Gutschi, Clemens et al. (2019). “Log-Based Predictive Maintenance in Discrete Parts Manufacturing”. In: *Procedia CIRP* 79, pp. 528–533. ISSN: 22128271. DOI: 10.1016/j.procir.2019.02.098. URL: https://linkinghub.elsevier.com/retrieve/pii/S221282711930215X (visited on 11/25/2023).

Harrando, Ismail et al. (2021). “Apples to Apples: A Systematic Evaluation of Topic Models”. In: *Proceedings of the Conference Recent Advances in Natural Language Processing - Deep Learning for Natural Language Processing Methods and Applications*. International Conference Recent Advances in Natural Language Processing. INCOMA Ltd. Shoumen, BULGARIA, pp. 483–493. ISBN: 978-954-452-072-4. DOI: 10.26615/978-954-452-072-4_055. URL: https://acl-bg.org/proceedings/2021/RANLP%202021/pdf/2021.ranlp-1.55.pdf (visited on 10/28/2023).

Hasan, Imran, Esmaeil Bahalkeh, and Yuehwern Yih (June 2020). “Evaluating Intensive Care Unit Admission and Discharge Policies Using a Discrete Event Simulation Model”. In: *SIMULATION* 96.6, pp. 501–518. ISSN: 0037-5497, 1741-3133. DOI: 10.1177/0037549720914749.

URL: http : / / journals . sagepub . com / doi / 10 . 1177 / 0037549720914749 (visited on 02/24/2024).

Hashemian, H. M. and Wendell C. Bean (Oct. 2011). "State-of-the-Art Predictive Maintenance Techniques*". In: *IEEE Transactions on Instrumentation and Measurement* 60.10, pp. 3480–3492. ISSN: 0018-9456, 1557-9662. DOI: 10.1109/TIM.2009.2036347. URL: http://ieeexplore.ieee.org/document/5754581/ (visited on 04/19/2024).

Hasterok, Constanze and Janina Stompe (Sept. 27, 2022). "PAISE ® – Process Model for AI Systems Engineering". In: *at - Automatisierungstechnik* 70.9, pp. 777–786. ISSN: 2196-677X, 0178-2312. DOI: 10.1515/auto-2022-0020. URL: https://www.degruyter.com/document/doi/10.1515/auto-2022-0020/html (visited on 10/15/2023).

He, Shilin et al. (Oct. 2016). "Experience Report: System Log Analysis for Anomaly Detection". In: *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*. 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE). Ottawa, ON, Canada: IEEE, pp. 207–218. ISBN: 978-1-4673-9002-6. DOI: 10.1109/ISSRE.2016.21. URL: http://ieeexplore.ieee.org/document/7774521/ (visited on 11/25/2023).

He, Xin, Kaiyong Zhao, and Xiaowen Chu (Jan. 2021). "AutoML: A Survey of the State-of-the-Art". In: *Knowledge-Based Systems* 212, p. 106622. ISSN: 09507051. DOI: 10 . 1016 / j . knosys . 2020 . 106622. URL: https : / / linkinghub . elsevier . com / retrieve / pii / S0950705120307516 (visited on 02/03/2024).

Heidrich, Benedikt et al. (2021). "pyWATTS: Python Workflow Automation Tool for Time Series". Version 1. In: DOI: 10.48550/ARXIV.2106.10157. URL: https://arxiv.org/abs/2106.10157 (visited on 02/03/2024).

Heineman, George T. and William T. Councill, eds. (2001). *Component-Based Software Engineering: Putting the Pieces Together*. Boston: Addison-Wesley. 818 pp. ISBN: 978-0-201-70485-3.

Herpel, Thomas et al. (June 2009). "Stochastic and Deterministic Performance Evaluation of Automotive CAN Communication". In: *Computer Networks* 53.8, pp. 1171–1185. ISSN: 13891286. DOI: 10.1016/j.comnet.2009.02.008. URL: https://linkinghub.elsevier.com/retrieve/pii/S1389128609000413 (visited on 11/25/2023).

Herrmann, Thomas (2020). "Socio-Technical Design of Hybrid Intelligence Systems – The Case of Predictive Maintenance". In: *Artificial Intelligence in HCI*. Ed. by Helmut Degen and Lauren Reinerman-Jones. Vol. 12217. Cham: Springer International Publishing, pp. 298–309. ISBN: 978-3-030-50333-8 978-3-030-50334-5. DOI: 10.1007/978-3-030-50334-5_20. URL: http://link.springer.com/10.1007/978-3-030-50334-5_20 (visited on 11/01/2023).

Hevner, Alan and Samir Chatterjee (2010). "Design Science Research in Information Systems". In: *Design Research in Information Systems*. Vol. 22. Boston, MA: Springer US, pp. 9–22. ISBN: 978-1-4419-5652-1 978-1-4419-5653-8. DOI: 10.1007/978-1-4419-5653-8_2. URL: http://link.springer.com/10.1007/978-1-4419-5653-8_2 (visited on 02/11/2021).

Hevner et al. (2004). "Design Science in Information Systems Research". In: *MIS Quarterly* 28.1, p. 75. ISSN: 02767783. DOI: 10.2307/25148625. JSTOR: 10.2307/25148625. URL: https://www.jstor.org/stable/10.2307/25148625 (visited on 02/11/2021).

Hewage, Nipuni and Dulani Meedeniya (2022). "Machine Learning Operations: A Survey on MLOps Tool Support". Version 2. In: DOI: 10.48550/ARXIV.2202.10169. URL: https://arxiv.org/abs/2202.10169 (visited on 11/01/2023).

Hoff, Gerd and Martin Mundhenk (Oct. 21, 2001). "Finding Scientific Papers with Homepagesearch and MOPS". In: *Proceedings of the 19th Annual International Conference on Computer Documentation*. SIGDOC01: 19th International Conference on Systems Documentation. Sante Fe New Mexico USA: ACM, pp. 201–207. ISBN: 978-1-58113-295-3. DOI: 10 . 1145 / 501516 . 501556. URL: https : / / dl . acm . org / doi / 10 . 1145 / 501516 . 501556 (visited on 08/25/2023).

Homma, Toshimitsu and Andrea Saltelli (Apr. 1996). "Importance Measures in Global Sensitivity Analysis of Nonlinear Models". In: *Reliability Engineering & System Safety* 52.1, pp. 1–

17. ISSN: 09518320. DOI: 10.1016/0951-8320(96)00002-6. URL: https://linkinghub.elsevier.com/retrieve/pii/0951832096000026 (visited on 04/17/2024).

Hosamo, Haidar Hosamo, Henrik Kofoed Nielsen, et al. (Feb. 2023a). "Digital Twin Framework for Automated Fault Source Detection and Prediction for Comfort Performance Evaluation of Existing Non-Residential Norwegian Buildings". In: *Energy and Buildings* 281, p. 112732. ISSN: 03787788. DOI: 10.1016/j.enbuild.2022.112732. URL: https://linkinghub.elsevier.com/retrieve/pii/S0378778822009033 (visited on 08/25/2023).

— (June 2023b). "Improving Building Occupant Comfort through a Digital Twin Approach: A Bayesian Network Model and Predictive Maintenance Method". In: *Energy and Buildings* 288, p. 112992. ISSN: 03787788. DOI: 10.1016/j.enbuild.2023.112992. URL: https://linkinghub.elsevier.com/retrieve/pii/S0378778823002220 (visited on 08/25/2023).

Hosamo, Haidar Hosamo, Paul Ragnar Svennevig, et al. (Apr. 2022). "A Digital Twin Predictive Maintenance Framework of Air Handling Units Based on Automatic Fault Detection and Diagnostics". In: *Energy and Buildings* 261, p. 111988. ISSN: 03787788. DOI: 10.1016/j.enbuild.2022.111988. URL: https://linkinghub.elsevier.com/retrieve/pii/S0378778822001591 (visited on 08/25/2023).

Hu, Xing et al. (May 28, 2018). "Deep Code Comment Generation". In: *Proceedings of the 26th Conference on Program Comprehension*. ICSE '18: 40th International Conference on Software Engineering. Gothenburg Sweden: ACM, pp. 200–210. ISBN: 978-1-4503-5714-2. DOI: 10.1145/3196321.3196334. URL: https://dl.acm.org/doi/10.1145/3196321.3196334 (visited on 02/25/2024).

Iooss, Bertrand and Paul Lemaître (2015). "A Review on Global Sensitivity Analysis Methods". In: *Uncertainty Management in Simulation-Optimization of Complex Systems*. Ed. by Gabriella Dellino and Carlo Meloni. Vol. 59. Boston, MA: Springer US, pp. 101–122. ISBN: 978-1-4899-7546-1 978-1-4899-7547-8. DOI: 10.1007/978-1-4899-7547-8_5. URL: https://link.springer.com/10.1007/978-1-4899-7547-8_5 (visited on 04/17/2024).

Jahn, Markus et al. (Aug. 2012). "Supporting Model Maintenance in Component-based Product Lines". In: *2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture*. 2012 Joint Working IEEE/IFIP Conference on Software Architecture (WICSA) & European Conference on Software Architecture (ECSA). Helsinki, Finland: IEEE, pp. 21–30. ISBN: 978-1-4673-2809-8 978-0-7695-4827-2. DOI: 10.1109/WICSA-ECSA.212.10. URL: http://ieeexplore.ieee.org/document/6337758/ (visited on 01/19/2024).

Jamrozik, Anja and Dedre Gentner (Mar. 2020). "Relational Labeling Unlocks Inert Knowledge". In: *Cognition* 196, p. 104146. ISSN: 00100277. DOI: 10.1016/j.cognition.2019.104146. URL: https://linkinghub.elsevier.com/retrieve/pii/S0010027719303208 (visited on 11/05/2023).

Jardine, Andrew K.S., Daming Lin, and Dragan Banjevic (Oct. 2006). "A Review on Machinery Diagnostics and Prognostics Implementing Condition-Based Maintenance". In: *Mechanical Systems and Signal Processing* 20.7, pp. 1483–1510. ISSN: 08883270. DOI: 10.1016/j.ymssp.2005.09.012. URL: https://linkinghub.elsevier.com/retrieve/pii/S0888327005001512 (visited on 11/25/2023).

Jarzabek, S. and M. Hitz (1998). "Business-Oriented Component-Based Software Development and Evolution". In: *Proceedings Ninth International Workshop on Database and Expert Systems Applications (Cat. No.98EX130)*. Ninth International Workshop on Database and Expert Systems Applications. Vienna, Austria: IEEE Comput. Soc, pp. 784–788. ISBN: 978-0-8186-8353-4. DOI: 10.1109/DEXA.1998.707496. URL: http://ieeexplore.ieee.org/document/707496/ (visited on 01/19/2024).

Jin, Haifeng, Qingquan Song, and Xia Hu (2018). "Auto-Keras: An Efficient Neural Architecture Search System". Version 3. In: DOI: 10.48550/ARXIV.1806.10282. URL: https://arxiv.org/abs/1806.10282 (visited on 02/03/2024).

Jordan, M. I. and T. M. Mitchell (July 17, 2015). “Machine Learning: Trends, Perspectives, and Prospects”. In: *Science* 349.6245, pp. 255–260. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.aaa8415. URL: https://www.science.org/doi/10.1126/science.aaa8415 (visited on 11/01/2023).

Kanajan, S. et al. (2006). “Exploring Trade-off’s Between Centralized versus Decentralized Automotive Architectures Using a Virtual Integration Environment”. In: *Proceedings of the Design Automation & Test in Europe Conference.* 2006 Design, Automation and Test in Europe. Munich, Germany: IEEE, pp. 1–6. ISBN: 978-3-9810801-1-7. DOI: 10.1109/DATE.2006.243895. URL: http://ieeexplore.ieee.org/document/1656943/ (visited on 11/05/2023).

Kang, Youn-ah and John Stasko (Dec. 2012). “Examining the Use of a Visual Analytics System for Sensemaking Tasks: Case Studies with Domain Experts”. In: *IEEE Transactions on Visualization and Computer Graphics* 18.12, pp. 2869–2878. ISSN: 1077-2626. DOI: 10.1109/TVCG.2012.224. URL: http://ieeexplore.ieee.org/document/6327293/ (visited on 11/05/2023).

Karpatne, Anuj et al. (Oct. 1, 2017). “Theory-Guided Data Science: A New Paradigm for Scientific Discovery from Data”. In: *IEEE Transactions on Knowledge and Data Engineering* 29.10, pp. 2318–2331. ISSN: 1041-4347. DOI: 10.1109/TKDE.2017.2720168. URL: http://ieeexplore.ieee.org/document/7959606/ (visited on 03/02/2024).

Kathirvelkumaran, L. and Ravichandran Moorthy (Jan. 25, 2023). “An Architectural Pattern for Dynamic Component Integration”. In: *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, pp. 101–106. ISSN: 2456-3307. DOI: 10.32628/CSEIT228626. URL: https://ijsrcseit.com/CSEIT228626 (visited on 01/19/2024).

Kaur, Kulbir and Harshpreet Singh (May 2014). “PROMETHEE Based Component Evaluation and Selection for Component Based Software Engineering”. In: *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies.* 2014 International Conference on Advanced Communication, Control and Computing Technologies (ICACCCT). Ramanathapuram, India: IEEE, pp. 1421–1425. ISBN: 978-1-4799-3914-5 978-1-4799-3913-8. DOI: 10.1109/ICACCCT.2014.7019336. URL: http://ieeexplore.ieee.org/document/7019336/ (visited on 10/15/2023).

Khorsheed, Raghad M and Omer Faruk Beyca (Apr. 2021). “An Integrated Machine Learning: Utility Theory Framework for Real-Time Predictive Maintenance in Pumping Systems”. In: *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 235.5, pp. 887–901. ISSN: 0954-4054, 2041-2975. DOI: 10.1177/0954405420970517. URL: http://journals.sagepub.com/doi/10.1177/0954405420970517 (visited on 11/01/2023).

Kiangala, Kahiomba Sonia and Zenghui Wang (2020). “An Effective Predictive Maintenance Framework for Conveyor Motors Using Dual Time-Series Imaging and Convolutional Neural Network in an Industry 4.0 Environment”. In: *IEEE Access* 8, pp. 121033–121049. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3006788. URL: https://ieeexplore.ieee.org/document/9131796/ (visited on 11/01/2023).

Kim, Donghwan, Seungchul Lee, and Daeyoung Kim (June 2, 2021). “An Applicable Predictive Maintenance Framework for the Absence of Run-to-Failure Data”. In: *Applied Sciences* 11.11, p. 5180. ISSN: 2076-3417. DOI: 10.3390/app11115180. URL: https://www.mdpi.com/2076-3417/11/11/5180 (visited on 08/25/2023).

Kim, Yongbeom and Edward A. Stohr (Mar. 1998). “Software Reuse: Survey and Research Directions”. In: *Journal of Management Information Systems* 14.4, pp. 113–147. ISSN: 0742-1222, 1557-928X. DOI: 10.1080/07421222.1998.11518188. URL: https://www.tandfonline.com/doi/full/10.1080/07421222.1998.11518188 (visited on 04/19/2024).

Kitchenham, Barbara Ann (2004). “Procedures for Performing Systematic Reviews”. In: URL: https://api.semanticscholar.org/CorpusID:54019416.

Kong, In-Yeup (2022). “Study on Predictive Maintenance for Controller Failure”. In: *International Journal of Advanced Engineering, Management and Science* 8.5, pp. 01–05. ISSN: 24541311. DOI: 10.22161/ijaems.85.1. URL: https://ijaems.com/detail/study-on-predictive-maintenance-for-controller-failure/ (visited on 11/05/2023).

Koteska, Bojana and Goran Velinov (2013). “Component-Based Development: A Unified Model of Reusability Metrics”. In: *ICT Innovations 2012: Secure and Intelligent Systems.* Advances in Intelligent Systems and Computing 207. Berlin New York: Springer. ISBN: 978-3-642-37169-1.

Krueger, Charles W. (June 1992). “Software Reuse”. In: *ACM Computing Surveys* 24.2, pp. 131–183. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/130844.130856. URL: https://dl.acm.org/doi/10.1145/130844.130856 (visited on 11/25/2023).

Kumar, Deepak and Meghna Kumari (Sept. 2015). “Component Based Software Engineering: Quality Assurance Models, Metrics”. In: *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions).* 2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions). Noida, India: IEEE, pp. 1–6. ISBN: 978-1-4673-7231-2. DOI: 10.1109/ICRITO.2015.7359358. URL: http://ieeexplore.ieee.org/document/7359358/ (visited on 01/19/2024).

Kunzer, Brian, Mario Berges, and Artur Dubrawski (2022). “The Digital Twin Landscape at the Crossroads of Predictive Maintenance, Machine Learning and Physics Based Modeling”. Version 2. In: DOI: 10.48550/ARXIV.2206.10462. URL: https://arxiv.org/abs/2206.10462 (visited on 11/01/2023).

Larocque-Villiers, Justin, Patrick Dumond, and David Knox (Oct. 28, 2021). “Automating Predictive Maintenance Using State-Based Transfer Learning and Ensemble Methods”. In: *2021 IEEE International Symposium on Robotic and Sensors Environments (ROSE).* 2021 IEEE International Symposium on Robotic and Sensors Environments (ROSE). FL, USA: IEEE, pp. 1–7. ISBN: 978-1-66544-062-2. DOI: 10.1109/ROSE52750.2021.9611768. URL: https://ieeexplore.ieee.org/document/9611768/ (visited on 11/01/2023).

Lechuga, Yolanda et al. (Jan. 3, 2023). “Development of an Automated Design Tool for FEM-Based Characterization of Solid and Hollow Microneedles”. In: *Micromachines* 14.1, p. 133. ISSN: 2072-666X. DOI: 10.3390/mi14010133. URL: https://www.mdpi.com/2072-666X/14/1/133 (visited on 01/02/2024).

LeClair, Alexander, Sakib Haque, et al. (2020). “Improved Code Summarization via a Graph Neural Network”. Version 2. In: DOI: 10.48550/ARXIV.2004.02843. URL: https://arxiv.org/abs/2004.02843 (visited on 02/25/2024).

LeClair, Alexander, Siyuan Jiang, and Collin McMillan (2019). “A Neural Model for Generating Natural Language Summaries of Program Subroutines”. Version 1. In: DOI: 10.48550/ARXIV.1902.01954. URL: https://arxiv.org/abs/1902.01954 (visited on 02/25/2024).

Lee, Jay, Behrad Bagheri, and Hung-An Kao (Jan. 2015). “A Cyber-Physical Systems Architecture for Industry 4.0-Based Manufacturing Systems”. In: *Manufacturing Letters* 3, pp. 18–23. ISSN: 22138463. DOI: 10.1016/j.mfglet.2014.12.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S221384631400025X (visited on 08/25/2023).

Lee, Jay, Edzel Lapira, et al. (Oct. 2013). “Recent Advances and Trends in Predictive Manufacturing Systems in Big Data Environment”. In: *Manufacturing Letters* 1.1, pp. 38–41. ISSN: 22138463. DOI: 10.1016/j.mfglet.2013.09.005. URL: https://linkinghub.elsevier.com/retrieve/pii/S2213846313000114 (visited on 08/25/2023).

Lee, Jay, Fangji Wu, et al. (Jan. 2014). “Prognostics and Health Management Design for Rotary Machinery Systems—Reviews, Methodology and Applications”. In: *Mechanical Systems and Signal Processing* 42.1-2, pp. 314–334. ISSN: 08883270. DOI: 10.1016/j.ymssp.2013.06.004.

URL: https://linkinghub.elsevier.com/retrieve/pii/S0888327013002860 (visited on 08/24/2023).

Lee, Kangbae et al. (Apr. 30, 2017). "A Study on IoT-based Fleet Maintenance Management". In: *International Journal of Control and Automation* 10.4, pp. 287–296. ISSN: 20054297, 20054297. DOI: 10.14257/ijca.2017.10.4.25. URL: http://article.nadiapub.com/IJCA/vol10_no4/25.pdf (visited on 10/28/2023).

Lee, Mei-Ling Ting and G. A. Whitmore (Nov. 1, 2006). "Threshold Regression for Survival Analysis: Modeling Event Times by a Stochastic Process Reaching a Boundary". In: *Statistical Science* 21.4. ISSN: 0883-4237. DOI: 10.1214/088342306000000330. URL: https://projecteuclid.org/journals/statistical-science/volume-21/issue-4/Threshold-Regression-for-Survival-Analysis--Modeling-Event-Times-by/10.1214/088342306000000330.full (visited on 11/25/2023).

Lee, Sang M., DonHee Lee, and Youn Sung Kim (Dec. 2019). "The Quality Management Ecosystem for Predictive Maintenance in the Industry 4.0 Era". In: *International Journal of Quality Innovation* 5.1, p. 4. ISSN: 2363-7021. DOI: 10.1186/s40887-019-0029-5. URL: https://link.springer.com/10.1186/s40887-019-0029-5 (visited on 11/05/2023).

Lee, Young M. et al. (2007). "Discrete Event Simulation Modeling of Resource Planning and Service Order Execution for Service Businesses". In: *2007 Winter Simulation Conference*. 2007 Winter Simulation Conference. Washington, DC, USA: IEEE, pp. 2227–2233. ISBN: 978-1-4244-1305-8. DOI: 10.1109/WSC.2007.4419858. URL: http://ieeexplore.ieee.org/document/4419858/ (visited on 02/24/2024).

Lehman, M.M. and J.F. Ramil (2000). "Software Evolution in the Age of Component-Based Software Engineering". In: *IEE Proceedings - Software* 147.6, p. 249. ISSN: 14625970. DOI: 10.1049/ip-sen:20000922. URL: https://digital-library.theiet.org/content/journals/10.1049/ip-sen_20000922 (visited on 10/15/2023).

Levy, Yair and Timothy J. Ellis (2011). "A Guide for Novice Researchers on Experimental and Quasi-Experimental Studies in Information Systems Research". In: *Interdisciplinary Journal of Information, Knowledge, and Management* 6, pp. 151–161. ISSN: 1555-1229, 1555-1237. DOI: 10.28945/1373. URL: https://www.informingscience.org/Publications/1373 (visited on 11/25/2023).

Lewin, Kurt (Nov. 1946). "Action Research and Minority Problems". In: *Journal of Social Issues* 2.4, pp. 34–46. ISSN: 0022-4537, 1540-4560. DOI: 10.1111/j.1540-4560.1946.tb02295.x. URL: https://spssi.onlinelibrary.wiley.com/doi/10.1111/j.1540-4560.1946.tb02295.x (visited on 11/05/2023).

Li, Guoyan et al. (July 2021). "Data Science Skills and Domain Knowledge Requirements in the Manufacturing Industry: A Gap Analysis". In: *Journal of Manufacturing Systems* 60, pp. 692–706. ISSN: 02786125. DOI: 10.1016/j.jmsy.2021.07.007. URL: https://linkinghub.elsevier.com/retrieve/pii/S0278612521001448 (visited on 04/18/2024).

Li, Hongfei et al. (Aug. 2014). "Improving Rail Network Velocity: A Machine Learning Approach to Predictive Maintenance". In: *Transportation Research Part C: Emerging Technologies* 45, pp. 17–26. ISSN: 0968090X. DOI: 10.1016/j.trc.2014.04.013. URL: https://linkinghub.elsevier.com/retrieve/pii/S0968090X14001107 (visited on 10/27/2023).

Li, Xinyu et al. (Jan. 2021). "A Data-Driven Reversible Framework for Achieving Sustainable Smart Product-Service Systems". In: *Journal of Cleaner Production* 279, p. 123618. ISSN: 09596526. DOI: 10.1016/j.jclepro.2020.123618. URL: https://linkinghub.elsevier.com/retrieve/pii/S0959652620336635 (visited on 08/24/2023).

Lim, W.C. (Sept. 1994). "Effects of Reuse on Quality, Productivity, and Economics". In: *IEEE Software* 11.5, pp. 23–30. ISSN: 0740-7459. DOI: 10.1109/52.311048. URL: http://ieeexplore.ieee.org/document/311048/ (visited on 08/24/2023).

Liu, Yuehua et al. (Feb. 2022). "Empowering IoT Predictive Maintenance Solutions With AI: A Distributed System for Manufacturing Plant-Wide Monitoring". In: *IEEE Transactions* *on Industrial Informatics* 18.2, pp. 1345–1354. ISSN: 1551-3203, 1941-0050. DOI: 10.1109/TII.2021.3091774. URL: https://ieeexplore.ieee.org/document/9463585/ (visited on 01/02/2024).

Löffler, Elisabeth, Nicole Von Der Linden, and Wolfgang Schneider (Oct. 19, 2016). “Influence of Domain Knowledge on Monitoring Performance Across the Life Span”. In: *Journal of Cognition and Development* 17.5, pp. 765–785. ISSN: 1524-8372, 1532-7647. DOI: 10.1080/15248372.2016.1208204. URL: https://www.tandfonline.com/doi/full/10.1080/15248372.2016.1208204 (visited on 11/05/2023).

Loiret, Frédéric et al. (June 20, 2011). “Software Engineering of Component-Based Systems-of-Systems: A Reference Framework”. In: *Proceedings of the 14th International ACM Sigsoft Symposium on Component Based Software Engineering.* Comparch ’11: Federated Events on Component-Based Software Engineering and Software Architecture. Boulder Colorado USA: ACM, pp. 61–66. ISBN: 978-1-4503-0723-9. DOI: 10.1145/2000229.2000238. URL: https://dl.acm.org/doi/10.1145/2000229.2000238 (visited on 10/15/2023).

Lopinski, Juliane, Sabine Sachweh, and Claudia Müller (Sept. 7, 2023). “Innovating Smart Buildings and Cities Through Agent-Based Modeling, Discrete Event Simulation and Sensor-Based Data Analytics”. In: *2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS).* 2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS). Dortmund, Germany: IEEE, pp. 341–345. ISBN: 9798350358056. DOI: 10.1109/IDAACS58523.2023.10348759. URL: https://ieeexplore.ieee.org/document/10348759/ (visited on 02/24/2024).

Lv, Yaqiong et al. (Sept. 12, 2023). “A Predictive Maintenance Strategy for Multi-Component Systems Based on Components’ Remaining Useful Life Prediction”. In: *Mathematics* 11.18, p. 3884. ISSN: 2227-7390. DOI: 10.3390/math11183884. URL: https://www.mdpi.com/2227-7390/11/18/3884 (visited on 04/18/2024).

Lwakatare, Lucy Ellen et al. (Nov. 2020). “Large-Scale Machine Learning Systems in Real-World Industrial Settings: A Review of Challenges and Solutions”. In: *Information and Software Technology* 127, p. 106368. ISSN: 09505849. DOI: 10.1016/j.infsof.2020.106368. URL: https://linkinghub.elsevier.com/retrieve/pii/S0950584920301373 (visited on 11/01/2023).

Lyubchyk, Leonid et al. (Dec. 9, 2022). “Machine Learning-Based Failure Rate Identification for Predictive Maintenance in Industry 4.0”. In: *2022 12th International Conference on Dependable Systems, Services and Technologies (DESSERT).* 2022 12th International Conference on Dependable Systems, Services and Technologies (DESSERT). Athens, Greece: IEEE, pp. 1–5. ISBN: 9798350333046. DOI: 10.1109/DESSERT58054.2022.10018614. URL: https://ieeexplore.ieee.org/document/10018614/ (visited on 01/02/2024).

Maier, Daniel et al. (Apr. 3, 2018). “Applying LDA Topic Modeling in Communication Research: Toward a Valid and Reliable Methodology”. In: *Communication Methods and Measures* 12.2-3, pp. 93–118. ISSN: 1931-2458, 1931-2466. DOI: 10.1080/19312458.2018.1430754. URL: https://www.tandfonline.com/doi/full/10.1080/19312458.2018.1430754 (visited on 10/28/2023).

Makinen, Sasu et al. (May 2021). “Who Needs MLOps: What Data Scientists Seek to Accomplish and How Can MLOps Help?” In: *2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN).* 2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN). Madrid, Spain: IEEE, pp. 109–112. ISBN: 978-1-66544-470-5. DOI: 10.1109/WAIN52551.2021.00024. URL: https://ieeexplore.ieee.org/document/9474355/ (visited on 11/01/2023).

Mallela, Sree Naga Raja Sekhar et al. (Feb. 28, 2023). “MLOps and SecOPS Affecting Data Science”. In: *International Journal for Research in Applied Science and Engineering Technology* 11.2, pp. 217–222. ISSN: 23219653. DOI: 10.22214/ijraset.2023.48997. URL: https:

//www.ijraset.com/best-journal/mlops-and-secops-affecting-data-science (visited on 11/01/2023).

Mann, Lawrence, Anuj Saxena, and Gerald M. Knapp (Mar. 1, 1995). “Statistical-based or Condition-based Preventive Maintenance?” In: *Journal of Quality in Maintenance Engineering* 1.1, pp. 46–59. ISSN: 1355-2511. DOI: 10.1108/13552519510083156. URL: https://www.emerald.com/insight/content/doi/10.1108/13552519510083156/full/html (visited on 11/25/2023).

March, Salvatore T. and Gary D. Scudder (Apr. 2019). “Predictive Maintenance: Strategic Use of IT in Manufacturing Organizations”. In: *Information Systems Frontiers* 21.2, pp. 327–341. ISSN: 1387-3326, 1572-9419. DOI: 10.1007/s10796-017-9749-z. URL: http://link.springer.com/10.1007/s10796-017-9749-z (visited on 10/28/2023).

March, Salvatore T. and Gerald F. Smith (Dec. 1995). “Design and Natural Science Research on Information Technology”. In: *Decision Support Systems* 15.4, pp. 251–266. ISSN: 01679236. DOI: 10.1016/0167-9236(94)00041-2. URL: https://linkinghub.elsevier.com/retrieve/pii/0167923694000412 (visited on 11/05/2023).

Mboweni, Tsakani, Themba Masombuka, and Cyrille Dongmo (July 20, 2022). “A Systematic Review of Machine Learning DevOps”. In: *2022 International Conference on Electrical, Computer and Energy Technologies (ICECET).* 2022 International Conference on Electrical, Computer and Energy Technologies (ICECET). Prague, Czech Republic: IEEE, pp. 1–6. ISBN: 978-1-66547-087-2. DOI: 10.1109/ICECET55527.2022.9872968. URL: https://ieeexplore.ieee.org/document/9872968/ (visited on 11/01/2023).

McCabe, T.J. (Dec. 1976). “A Complexity Measure”. In: *IEEE Transactions on Software Engineering* SE-2.4, pp. 308–320. ISSN: 0098-5589. DOI: 10.1109/TSE.1976.233837. URL: http://ieeexplore.ieee.org/document/1702388/ (visited on 01/16/2024).

Meho, Lokman I. and Yvonne Rogers (Sept. 2008). “Citation Counting, Citation Ranking, and *h* -Index of Human-Computer Interaction Researchers: A Comparison of Scopus and Web of Science”. In: *Journal of the American Society for Information Science and Technology* 59.11, pp. 1711–1726. ISSN: 15322882, 15322890. DOI: 10.1002/asi.20874. URL: https://onlinelibrary.wiley.com/doi/10.1002/asi.20874 (visited on 08/25/2023).

Mi, Shanghua et al. (Jan. 2021). “Prediction Maintenance Integrated Decision-Making Approach Supported by Digital Twin-Driven Cooperative Awareness and Interconnection Framework”. In: *Journal of Manufacturing Systems* 58, pp. 329–345. ISSN: 02786125. DOI: 10.1016/j.jmsy.2020.08.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S0278612520301345 (visited on 08/24/2023).

Mili, Hafedh (2002). *Reuse Based Software Engineering: Techniques, Organization and Measurement.* New York: Wiley. 636 pp. ISBN: 978-0-471-39819-6.

Mimno, David et al. (2011). “Optimizing Semantic Coherence in Topic Models”. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* EMNLP ’11. USA: Association for Computational Linguistics, pp. 262–272. ISBN: 978-1-937284-11-4.

Mitchell, Andrew A., J.Edward Russo, and Dick R. Wittink (Apr. 1991). “Issues in the Development and Use of Expert Systems for Marketing Decisions”. In: *International Journal of Research in Marketing* 8.1, pp. 41–50. ISSN: 01678116. DOI: 10.1016/0167-8116(91)90006-S. URL: https://linkinghub.elsevier.com/retrieve/pii/016781169190006S (visited on 11/05/2023).

Mobley, R. Keith (2002a). *An Introduction to Predictive Maintenance.* 2nd ed. Amsterdam ; New York: Butterworth-Heinemann. 438 pp. ISBN: 978-0-7506-7531-4.

— (2002b). *An Introduction to Predictive Maintenance.* 2nd ed. Amsterdam ; New York: Butterworth-Heinemann. 438 pp. ISBN: 978-0-7506-7531-4.

Mongeon, Philippe and Adèle Paul-Hus (Jan. 2016). “The Journal Coverage of Web of Science and Scopus: A Comparative Analysis”. In: *Scientometrics* 106.1, pp. 213–228. ISSN: 0138-9130, 1588-2861. DOI: 10.1007/s11192-015-1765-5. URL: http://link.springer.com/10.1007/s11192-015-1765-5 (visited on 08/25/2023).

Morstatter, Fred and Huan Liu (2016). "A Novel Measure for Coherence in Statistical Topic Models". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Berlin, Germany: Association for Computational Linguistics, pp. 543–548. DOI: 10.18653/v1/P16-2088. URL: http://aclweb.org/anthology/P16-2088 (visited on 08/27/2023).

Moser, Karin S. (Oct. 2017). "The Influence of Feedback and Expert Status in Knowledge Sharing Dilemmas". In: *Applied Psychology* 66.4, pp. 674–709. ISSN: 0269-994X, 1464-0597. DOI: 10.1111/apps.12105. URL: https://iaap-journals.onlinelibrary.wiley.com/doi/10.1111/apps.12105 (visited on 11/05/2023).

Mueller, Frank and Romano Dyerson (Mar. 1999). "Expert Humans or Expert Organizations?" In: *Organization Studies* 20.2, pp. 225–256. ISSN: 0170-8406, 1741-3044. DOI: 10.1177/0170840699202003. URL: http://journals.sagepub.com/doi/10.1177/0170840699202003 (visited on 11/05/2023).

Naskos, Athanasios et al. (2019). "Detecting Anomalous Behavior Towards Predictive Maintenance". In: *Advanced Information Systems Engineering Workshops*. Ed. by Henderik A. Proper and Janis Stirna. Vol. 349. Cham: Springer International Publishing, pp. 73–82. ISBN: 978-3-030-20947-6 978-3-030-20948-3. DOI: 10.1007/978-3-030-20948-3_7. URL: http://link.springer.com/10.1007/978-3-030-20948-3_7 (visited on 08/25/2023).

Nonaka, Y. et al. (Aug. 2015). "The S-Model: A Digital Manufacturing System Combined with Autonomous Statistical Analysis and Autonomous Discrete-Event Simulation for Smart Manufacturing". In: *2015 IEEE International Conference on Automation Science and Engineering (CASE)*. 2015 IEEE International Conference on Automation Science and Engineering (CASE). Gothenburg, Sweden: IEEE, pp. 1006–1011. ISBN: 978-1-4673-8183-3. DOI: 10.1109/CoASE.2015.7294230. URL: http://ieeexplore.ieee.org/document/7294230/ (visited on 02/24/2024).

Nunes, P., J. Santos, and E. Rocha (Feb. 2023). "Challenges in Predictive Maintenance – A Review". In: *CIRP Journal of Manufacturing Science and Technology* 40, pp. 53–67. ISSN: 17555817. DOI: 10.1016/j.cirpj.2022.11.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S1755581722001742 (visited on 04/18/2024).

O'Keefe, Robert M. (Feb. 1985). "Expert Systems and Operational Research-Mutual Benefits". In: *Journal of the Operational Research Society* 36.2, pp. 125–129. ISSN: 0160-5682, 1476-9360. DOI: 10.1057/jors.1985.25. URL: https://www.tandfonline.com/doi/full/10.1057/jors.1985.25 (visited on 11/05/2023).

O'Callaghan, Derek et al. (Aug. 2015). "An Analysis of the Coherence of Descriptors in Topic Modeling". In: *Expert Systems with Applications* 42.13, pp. 5645–5657. ISSN: 09574174. DOI: 10.1016/j.eswa.2015.02.055. URL: https://linkinghub.elsevier.com/retrieve/pii/S0957417415001633 (visited on 08/28/2023).

Olivier, Martin and Wouter Verschoof-van Der Vaart (Dec. 8, 2021). "Implementing State-of-the-Art Deep Learning Approaches for Archaeological Object Detection in Remotely-Sensed Data: The Results of Cross-Domain Collaboration". In: *Journal of Computer Applications in Archaeology* 4.1, pp. 274–289. ISSN: 2514-8362. DOI: 10.5334/jcaa.78. URL: http://journal.caa-international.org/articles/10.5334/jcaa.78/ (visited on 11/05/2023).

Omar, Muhammad et al. (Oct. 2015). "LDA Topics: Representation and Evaluation". In: *Journal of Information Science* 41.5, pp. 662–675. ISSN: 0165-5515, 1741-6485. DOI: 10.1177/0165551515587839. URL: http://journals.sagepub.com/doi/10.1177/0165551515587839 (visited on 08/27/2023).

Ong, Kevin Shen Hoong et al. (Apr. 1, 2022). "Deep-Reinforcement-Learning-Based Predictive Maintenance Model for Effective Resource Management in Industrial IoT". In: *IEEE Internet* *of Things Journal* 9.7, pp. 5173–5188. ISSN: 2327-4662, 2372-2541. DOI: 10.1109/JIOT.2021.3109955. URL: https://ieeexplore.ieee.org/document/9528837/ (visited on 11/01/2023).

Orso, A. et al. (2001). “Using Component Metacontent to Support the Regression Testing of Component-Based Software”. In: *Proceedings IEEE International Conference on Software Maintenance. ICSM 2001.* IEEE International Conference on Software Maintenance. ICSM 2001. Florence, Italy: IEEE Comput. Soc, pp. 716–725. ISBN: 978-0-7695-1189-4. DOI: 10.1109/ICSM.2001.972790. URL: http://ieeexplore.ieee.org/document/972790/ (visited on 01/19/2024).

Othmane, Lotfi ben, Martin Gilje Jaatun, and Edgar R. Weippl (2018). *Empirical Research for Software Security: Foundations and Experience.* First edition. Boca Raton, FL: CRC Press. ISBN: 978-1-315-15485-5.

Paganelli, Matteo et al. (Sept. 2020). “Explaining Data with Descriptions”. In: *Information Systems* 92, p. 101549. ISSN: 03064379. DOI: 10.1016/j.is.2020.101549. URL: https://linkinghub.elsevier.com/retrieve/pii/S0306437920300508 (visited on 03/02/2024).

Pan, M. Ch., P. Sas, and H. Van Brussel (Oct. 2003). “Machine Condition Monitoring Using Signal Classification Techniques”. In: *Journal of Vibration and Control* 9.10, pp. 1103–1120. ISSN: 1077-5463, 1741-2986. DOI: 10.1177/107754603030683. URL: http://journals.sagepub.com/doi/10.1177/107754603030683 (visited on 11/25/2023).

Park, Soya et al. (Apr. 14, 2021). “Facilitating Knowledge Sharing from Domain Experts to Data Scientists for Building NLP Models”. In: *26th International Conference on Intelligent User Interfaces.* IUI ’21: 26th International Conference on Intelligent User Interfaces. College Station TX USA: ACM, pp. 585–596. ISBN: 978-1-4503-8017-1. DOI: 10.1145/3397481.3450637. URL: https://dl.acm.org/doi/10.1145/3397481.3450637 (visited on 11/05/2023).

Parnas, D. L. (Dec. 1972). “On the Criteria to Be Used in Decomposing Systems into Modules”. In: *Communications of the ACM* 15.12, pp. 1053–1058. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/361598.361623. URL: https://dl.acm.org/doi/10.1145/361598.361623 (visited on 08/24/2023).

Paszke, Adam et al. (2019). “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. Version 1. In: DOI: 10.48550/ARXIV.1912.01703. URL: https://arxiv.org/abs/1912.01703 (visited on 02/03/2024).

Pathan, A.S.K., Hyung-Woo Lee, and Choong Seon Hong (2006). “Security in Wireless Sensor Networks: Issues and Challenges”. In: *2006 8th International Conference Advanced Communication Technology.* 8th International Conference on Advanced Communication Technology. Phoenix Park, Korea: IEEE, 6 pp.–1048. ISBN: 978-89-5519-129-5. DOI: 10.1109/ICACT.2006.206151. URL: http://ieeexplore.ieee.org/document/1625756/ (visited on 11/05/2023).

Patil, Ravindra B et al. (July 2017). “Predictive Modeling for Corrective Maintenance of Imaging Devices from Machine Logs”. In: *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC).* 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). Seogwipo: IEEE, pp. 1676–1679. ISBN: 978-1-5090-2809-2. DOI: 10.1109/EMBC.2017.8037163. URL: https://ieeexplore.ieee.org/document/8037163/ (visited on 11/25/2023).

Pedregosa, Fabian et al. (2012). “Scikit-Learn: Machine Learning in Python”. Version 4. In: DOI: 10.48550/ARXIV.1201.0490. URL: https://arxiv.org/abs/1201.0490 (visited on 02/03/2024).

Peffers, Ken et al. (Dec. 2007). “A Design Science Research Methodology for Information Systems Research”. In: *Journal of Management Information Systems* 24.3, pp. 45–77. ISSN: 0742-1222, 1557-928X. DOI: 10.2753/MIS0742-1222240302. URL: https://www.tandfonline.com/doi/full/10.2753/MIS0742-1222240302 (visited on 02/11/2021).

Pennock, Michael J. and Douglas A. Bodner (July 2020). “A Methodology for Modeling Sociotechnical Systems to Facilitate Exploratory Policy Analysis”. In: *Systems Engineering*

23.4, pp. 409–422. ISSN: 1098-1241, 1520-6858. DOI: 10.1002/sys.21534. URL: https://incose.onlinelibrary.wiley.com/doi/10.1002/sys.21534 (visited on 04/13/2024).

Pereira, Paul Christopher (May 4, 2020). "Hidden Value in Maintenance System Data: Using Machine Learning to Correlate and Predict the Risk of Asset Failures". In: *Day 3 Wed, May 06, 2020*. Offshore Technology Conference. Houston, Texas, USA: OTC, D031S037R007. DOI: 10.4043/30730-MS. URL: https://onepetro.org/OTCONF/proceedings/20OTC/3-20OTC/Houston,%20Texas,%20USA/107413 (visited on 01/03/2024).

Perner, Petra, ed. (2019). *Advances in Data Mining: Applications and Theoretical Aspects: 19th Industrial Conference, ICDM 2019, New York, NY, USA, July 17-21, 2019: Proceedings*. Leipzig, Germany: ibai-publishing. 348 pp. ISBN: 978-3-942952-60-6.

Pölöskei, István (Apr. 9, 2021). "MLOps Approach in the Cloud-Native Data Pipeline Design". In: *Acta Technica Jaurinensis* 15.1, pp. 1–6. ISSN: 2064-5228. DOI: 10.14513/actatechjaur.00581. URL: https://acta.sze.hu/index.php/acta/article/view/581 (visited on 11/01/2023).

Poor, P., J. Basl, and D. Zenisek (Mar. 2019). "Predictive Maintenance 4.0 as next Evolution Step in Industrial Maintenance Development". In: *2019 International Research Conference on Smart Computing and Systems Engineering (SCSE)*. 2019 International Research Conference on Smart Computing and Systems Engineering (SCSE). Colombo, Sri Lanka: IEEE, pp. 245–253. ISBN: 978-955-704-121-6. DOI: 10.23919/SCSE.2019.8842659. URL: https://ieeexplore.ieee.org/document/8842659/ (visited on 08/24/2023).

Pophaley, Mahesh and R. K. Vyas (Jan. 2010). "Choice Criteria for Maintenance Strategy in Automotive Industries". In: *International Journal of Management Science and Engineering Management* 5.6, pp. 446–452. ISSN: 1750-9653, 1750-9661. DOI: 10.1080/17509653.2010.10671136. URL: https://www.tandfonline.com/doi/full/10.1080/17509653.2010.10671136 (visited on 08/24/2023).

Porter, Michael E and James E Heppelmann (2015). "How Smart, Connected Products Are Transforming Companies". In:

Poulin, J. S., J. M. Caruso, and D. R. Hancock (1993). "The Business Case for Software Reuse". In: *IBM Systems Journal* 32.4, pp. 567–594. ISSN: 0018-8670. DOI: 10.1147/sj.324.0567. URL: http://ieeexplore.ieee.org/document/5387336/ (visited on 01/16/2024).

Pour, G. (1998). "Towards Component-Based Software Engineering". In: *Proceedings. The Twenty-Second Annual International Computer Software and Applications Conference (Compsac '98) (Cat. No.98CB 36241)*. Proceedings. The Twenty-Second Annual International Computer Software and Applications Conference (Compsac '98) (Cat. No.98CB 36241). Vienna, Austria: IEEE Comput. Soc, p. 599. ISBN: 978-0-8186-8585-9. DOI: 10.1109/CMPSAC.1998.716732. URL: http://ieeexplore.ieee.org/document/716732/ (visited on 10/15/2023).

Prytz, Rune (2014). "Machine Learning Methods for Vehicle Predictive Maintenance Using Off-Board and on-Board Data". Halmstad: Halmstad University. ISBN: 9789187045189.

Rajkumar, Ragunathan (Raj) et al. (June 13, 2010). "Cyber-Physical Systems: The next Computing Revolution". In: *Proceedings of the 47th Design Automation Conference*. DAC '10: The 47th Annual Design Automation Conference 2010. Anaheim California: ACM, pp. 731–736. ISBN: 978-1-4503-0002-5. DOI: 10.1145/1837274.1837461. URL: https://dl.acm.org/doi/10.1145/1837274.1837461 (visited on 11/05/2023).

Ramirez-Marquez, Jose Emmanuel and Brian J. Sauser (Aug. 2009). "System Development Planning via System Maturity Optimization". In: *IEEE Transactions on Engineering Management* 56.3, pp. 533–548. ISSN: 0018-9391. DOI: 10.1109/TEM.2009.2013830. URL: http://ieeexplore.ieee.org/document/4806066/ (visited on 08/24/2023).

Randall, Robert Bond (Jan. 21, 2011). *Vibration-based Condition Monitoring: Industrial, Aerospace and Automotive Applications*. 1st ed. Wiley. ISBN: 978-0-470-74785-8 978-0-470-97766-8. DOI: 10.1002/9780470977668. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/9780470977668 (visited on 11/25/2023).

Ravichandar, D. (2022). *Maintenance for Profitability*. Blue Rose Publishers. URL: https://books.google.de/books?id=pXZ8EAAAQBAJ.

Razali, Muhammad Najib et al. (May 31, 2020). “Big Data Analytics for Predictive Maintenance in Maintenance Management”. In: *Property Management* 38.4, pp. 513–529. ISSN: 0263-7472. DOI: 10.1108/PM-12-2019-0070. URL: https://www.emerald.com/insight/content/doi/10.1108/PM-12-2019-0070/full/html (visited on 11/05/2023).

Reason, Peter and Hilary Bradbury, eds. (2001). *Handbook of Action Research: Participative Inquiry and Practice*. London ; Thousand Oaks, Calif: SAGE. 468 pp. ISBN: 978-0-7619-6645-6.

Ren, Huanhuan, Yingnan Liu, and Shupeng Zhang (Dec. 2022). “Research on After-Sales Information System And Data Of Automotive OEMs”. In: *2022 Euro-Asia Conference on Frontiers of Computer Science and Information Technology (FCSIT)*. 2022 Euro-Asia Conference on Frontiers of Computer Science and Information Technology (FCSIT). Beijing, China: IEEE, pp. 187–190. ISBN: 978-1-66546-353-9. DOI: 10.1109/FCSIT57414.2022.00046. URL: https://ieeexplore.ieee.org/document/10098274/ (visited on 11/05/2023).

Renggli, Cedric et al. (2021). “A Data Quality-Driven View of MLOps”. Version 1. In: DOI: 10.48550/ARXIV.2102.07750. URL: https://arxiv.org/abs/2102.07750 (visited on 11/01/2023).

Rezig, El Kindi et al. (Aug. 2020). “Debugging Large-Scale Data Science Pipelines Using Dagger”. In: *Proceedings of the VLDB Endowment* 13.12, pp. 2993–2996. ISSN: 2150-8097. DOI: 10.14778/3415478.3415527. URL: https://dl.acm.org/doi/10.14778/3415478.3415527 (visited on 03/02/2024).

Ritchey, T (July 2006). “Problem Structuring Using Computer-Aided Morphological Analysis”. In: *Journal of the Operational Research Society* 57.7, pp. 792–801. ISSN: 0160-5682, 1476-9360. DOI: 10.1057/palgrave.jors.2602177. URL: https://www.tandfonline.com/doi/full/10.1057/palgrave.jors.2602177 (visited on 04/14/2024).

Röder, Michael, Andreas Both, and Alexander Hinneburg (Feb. 2, 2015). “Exploring the Space of Topic Coherence Measures”. In: *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. WSDM 2015: Eighth ACM International Conference on Web Search and Data Mining. Shanghai China: ACM, pp. 399–408. ISBN: 978-1-4503-3317-7. DOI: 10.1145/2684822.2685324. URL: https://dl.acm.org/doi/10.1145/2684822.2685324 (visited on 08/28/2023).

Rodič, Blaž (Aug. 1, 2017). “Industry 4.0 and the New Simulation Modelling Paradigm”. In: *Organizacija* 50.3, pp. 193–207. ISSN: 1581-1832. DOI: 10.1515/orga-2017-0017. URL: https://www.sciendo.com/article/10.1515/orga-2017-0017 (visited on 08/24/2023).

Rögnvaldsson, Thorsteinn et al. (Mar. 2018). “Self-Monitoring for Maintenance of Vehicle Fleets”. In: *Data Mining and Knowledge Discovery* 32.2, pp. 344–384. ISSN: 1384-5810, 1573-756X. DOI: 10.1007/s10618-017-0538-6. URL: http://link.springer.com/10.1007/s10618-017-0538-6 (visited on 11/25/2023).

Rokhforoz, Pegah and Olga Fink (Sept. 2021). “Hierarchical Multi-Agent Predictive Maintenance Scheduling for Trains Using Price-Based Approach”. In: *Computers & Industrial Engineering* 159, p. 107475. ISSN: 03608352. DOI: 10.1016/j.cie.2021.107475. URL: https://linkinghub.elsevier.com/retrieve/pii/S036083522100379X (visited on 10/27/2023).

Rosati, Riccardo et al. (Jan. 2023). “From Knowledge-Based to Big Data Analytic Model: A Novel IoT and Machine Learning Based Decision Support System for Predictive Maintenance in Industry 4.0”. In: *Journal of Intelligent Manufacturing* 34.1, pp. 107–121. ISSN: 0956-5515, 1572-8145. DOI: 10.1007/s10845-022-01960-x. URL: https://link.springer.com/10.1007/s10845-022-01960-x (visited on 01/02/2024).

Rose, Michael E. and John R. Kitchin (July 2019). “Pybliometrics: Scriptable Bibliometrics Using a Python Interface to Scopus”. In: *SoftwareX* 10, p. 100263. ISSN: 23527110. DOI: 10.1016/j.softx.2019.100263. URL: https://linkinghub.elsevier.com/retrieve/pii/S2352711019300573 (visited on 08/25/2023).

Roux, Dirk J. et al. (2006). “Bridging the Science&#8211;Management Divide: Moving from Unidirectional Knowledge Transfer to Knowledge Interfacing and Sharing”. In: *Ecology and Society* 11.1, art4. ISSN: 1708-3087. DOI: 10.5751/ES-01643-110104. URL: http://www.ecologyandsociety.org/vol11/iss1/art4/ (visited on 11/05/2023).

Ruiz, Angie and Jose Guevara (Jan. 24, 2020). “Sustainable Decision-Making in Road Development: Analysis of Road Preservation Policies”. In: *Sustainability* 12.3, p. 872. ISSN: 2071-1050. DOI: 10.3390/su12030872. URL: https://www.mdpi.com/2071-1050/12/3/872 (visited on 08/24/2023).

Runeson, Per and Martin Höst (Apr. 2009). “Guidelines for Conducting and Reporting Case Study Research in Software Engineering”. In: *Empirical Software Engineering* 14.2, pp. 131–164. ISSN: 1382-3256, 1573-7616. DOI: 10.1007/s10664-008-9102-8. URL: http://link.springer.com/10.1007/s10664-008-9102-8 (visited on 10/15/2023).

Sadrani, Mohammad et al. (Oct. 2023). “Charging Strategy Selection for Electric Bus Systems: A Multi-Criteria Decision-Making Approach”. In: *Applied Energy* 347, p. 121415. ISSN: 03062619. DOI: 10.1016/j.apenergy.2023.121415. URL: https://linkinghub.elsevier.com/retrieve/pii/S0306261923007791 (visited on 04/25/2024).

Salierno, Giulio et al. (2020). “An Architecture for Predictive Maintenance of Railway Points Based on Big Data Analytics”. In: *Advanced Information Systems Engineering Workshops.* Ed. by Sophie Dupuy-Chessa and Henderik A. Proper. Vol. 382. Cham: Springer International Publishing, pp. 29–40. ISBN: 978-3-030-49164-2 978-3-030-49165-9. DOI: 10.1007/978-3-030-49165-9_3. URL: http://link.springer.com/10.1007/978-3-030-49165-9_3 (visited on 01/02/2024).

Saltelli, Andrea (May 2002). “Making Best Use of Model Evaluations to Compute Sensitivity Indices”. In: *Computer Physics Communications* 145.2, pp. 280–297. ISSN: 00104655. DOI: 10.1016/S0010-4655(02)00280-1. URL: https://linkinghub.elsevier.com/retrieve/pii/S0010465502002801 (visited on 04/16/2024).

Saltelli, Andrea, Paola Annoni, et al. (Feb. 2010). “Variance Based Sensitivity Analysis of Model Output. Design and Estimator for the Total Sensitivity Index”. In: *Computer Physics Communications* 181.2, pp. 259–270. ISSN: 00104655. DOI: 10.1016/j.cpc.2009.09.018. URL: https://linkinghub.elsevier.com/retrieve/pii/S0010465509003087 (visited on 04/17/2024).

Saltelli, Andrea, Marco Ratto, et al. (Dec. 18, 2007). *Global Sensitivity Analysis. The Primer.* 1st ed. Wiley. ISBN: 978-0-470-05997-5 978-0-470-72518-4. DOI: 10.1002/9780470725184. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/9780470725184 (visited on 04/17/2024).

Sambasivan, Nithya and Rajesh Veeraraghavan (Apr. 29, 2022). “The Deskilling of Domain Expertise in AI Development”. In: *CHI Conference on Human Factors in Computing Systems.* CHI ’22: CHI Conference on Human Factors in Computing Systems. New Orleans LA USA: ACM, pp. 1–14. ISBN: 978-1-4503-9157-3. DOI: 10.1145/3491102.3517578. URL: https://dl.acm.org/doi/10.1145/3491102.3517578 (visited on 11/05/2023).

Sametinger, Johannes (1997). *Software Engineering with Reusable Components.* Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-642-08299-3 978-3-662-03345-6. DOI: 10.1007/978-3-662-03345-6. URL: http://link.springer.com/10.1007/978-3-662-03345-6 (visited on 04/19/2024).

Sein et al. (2011). “Action Design Research”. In: *MIS Quarterly* 35.1, p. 37. ISSN: 02767783. DOI: 10.2307/23043488. JSTOR: 10.2307/23043488. URL: https://www.jstor.org/stable/10.2307/23043488 (visited on 11/05/2023).

Selçuk, Sahan Yoruç et al. (Sept. 22, 2021). “A Workflow for Synthetic Data Generation and Predictive Maintenance for Vibration Data”. In: *Information* 12.10, p. 386. ISSN: 2078-2489. DOI: 10.3390/info12100386. URL: https://www.mdpi.com/2078-2489/12/10/386 (visited on 01/02/2024).

Serna, Felix et al. (Sept. 2011). "&#x201C;Predictive Maintenance surveyor&#x201D; Design Pattern for Machine Tools Control Software Applications". In: *ETFA2011*. Factory Automation (ETFA 2011). Toulouse, France: IEEE, pp. 1–7. ISBN: 978-1-4577-0017-0. DOI: 10.1109/ETFA.2011.6059119. URL: http://ieeexplore.ieee.org/document/6059119/ (visited on 11/25/2023).

Sfeir, Georges, Constantinos Antoniou, and Nivine Abbas (Dec. 2018). "Simulation-Based Evacuation Planning Using State-of-the-Art Sensitivity Analysis Techniques". In: *Simulation Modelling Practice and Theory* 89, pp. 160–174. ISSN: 1569190X. DOI: 10.1016/j.simpat.2018.09.017. URL: https://linkinghub.elsevier.com/retrieve/pii/S1569190X1830145X (visited on 02/23/2024).

Shamayleh, Abdulrahim, Mahmoud Awad, and Jumana Farhat (Apr. 2020). "IoT Based Predictive Maintenance Management of Medical Equipment". In: *Journal of Medical Systems* 44.4, p. 72. ISSN: 0148-5598, 1573-689X. DOI: 10.1007/s10916-020-1534-8. URL: http://link.springer.com/10.1007/s10916-020-1534-8 (visited on 11/01/2023).

Shanin, Ivan, Sergey Stupnikov, and Viktor Zakharov (Sept. 2019). "Application of Anomaly Detection Methods in the Housing and Utility Infrastructure Data". In: *2019 Ivannikov Memorial Workshop (IVMEM)*. 2019 Ivannikov Memorial Workshop (IVMEM). Velikiy Novgorod, Russia: IEEE, pp. 101–105. ISBN: 978-1-72814-623-2. DOI: 10.1109/IVMEM.2019.00023. URL: https://ieeexplore.ieee.org/document/8880751/ (visited on 01/02/2024).

Shimada, Junya and Satoko Sakajo (July 2016). "A Statistical Approach to Reduce Failure Facilities Based on Predictive Maintenance". In: *2016 International Joint Conference on Neural Networks (IJCNN)*. 2016 International Joint Conference on Neural Networks (IJCNN). Vancouver, BC, Canada: IEEE, pp. 5156–5160. ISBN: 978-1-5090-0620-5. DOI: 10.1109/IJCNN.2016.7727880. URL: http://ieeexplore.ieee.org/document/7727880/ (visited on 11/25/2023).

Si, Xiao-Sheng et al. (Aug. 2011). "Remaining Useful Life Estimation – A Review on the Statistical Data Driven Approaches". In: *European Journal of Operational Research* 213.1, pp. 1–14. ISSN: 03772217. DOI: 10.1016/j.ejor.2010.11.018. URL: https://linkinghub.elsevier.com/retrieve/pii/S0377221710007903 (visited on 11/25/2023).

Siau, Keng and Matti Rossi (May 2011). "Evaluation Techniques for Systems Analysis and Design Modelling Methods - a Review and Comparative Analysis: Evaluation Techniques for Modelling Methods". In: *Information Systems Journal* 21.3, pp. 249–268. ISSN: 13501917. DOI: 10.1111/j.1365-2575.2007.00255.x. URL: https://onlinelibrary.wiley.com/doi/10.1111/j.1365-2575.2007.00255.x (visited on 04/14/2024).

Sikorska, J.Z., M. Hodkiewicz, and L. Ma (July 2011). "Prognostic Modelling Options for Remaining Useful Life Estimation by Industry". In: *Mechanical Systems and Signal Processing* 25.5, pp. 1803–1836. ISSN: 08883270. DOI: 10.1016/j.ymssp.2010.11.018. URL: https://linkinghub.elsevier.com/retrieve/pii/S0888327010004218 (visited on 11/25/2023).

Silvestrin, Luis P., Mark Hoogendoorn, and Ger Koole (Dec. 2019). "A Comparative Study of State-of-the-Art Machine Learning Algorithms for Predictive Maintenance". In: *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2019 IEEE Symposium Series on Computational Intelligence (SSCI). Xiamen, China: IEEE, pp. 760–767. ISBN: 978-1-72812-485-8. DOI: 10.1109/SSCI44817.2019.9003044. URL: https://ieeexplore.ieee.org/document/9003044/ (visited on 11/01/2023).

Simon, Herbert Alexander (2008). *The Sciences of the Artificial*. 3. ed., [Nachdr.] Cambridge, Mass.: MIT Press. 231 pp. ISBN: 978-0-262-69191-8 978-0-262-19374-0.

Sipos, Ruben et al. (Aug. 24, 2014). "Log-Based Predictive Maintenance". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '14: The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York New York USA: ACM, pp. 1867–1876. ISBN: 978-1-4503-2956-9. DOI:

10.1145/2623330.2623340. URL: https://dl.acm.org/doi/10.1145/2623330.2623340 (visited on 11/25/2023).

Sobol, I.M. (Feb. 2001). “Global Sensitivity Indices for Nonlinear Mathematical Models and Their Monte Carlo Estimates”. In: *Mathematics and Computers in Simulation* 55.1-3, pp. 271–280. ISSN: 03784754. DOI: 10.1016/S0378-4754(00)00270-6. URL: https://linkinghub.elsevier.com/retrieve/pii/S0378475400002706 (visited on 04/16/2024).

Sommerville, Ian (2011). *Software Engineering*. 9th ed. Boston: Pearson. 773 pp. ISBN: 978-0-13-703515-1 978-0-13-705346-9.

Spiegel, Stephan et al. (2018). “Cost-Sensitive Learning for Predictive Maintenance”. Version 1. In: DOI: 10.48550/ARXIV.1809.10979. URL: https://arxiv.org/abs/1809.10979 (visited on 11/01/2023).

Spivey, J. M. (1989). *The Z Notation: A Reference Manual*. Prentice-Hall International Series in Computer Science. Englewood Cliffs, N.J: Prentice Hall. 155 pp. ISBN: 978-0-13-983768-5.

Stallinger, F. et al. (2002). “Software Process Improvement for Component-Based Software Engineering: An Introduction to the OOSPICE Project”. In: *Proceedings. 28th Euromicro Conference*. 28th Euromicro Conference. Dortmund, Germany: IEEE Comput. Soc, pp. 318–323. ISBN: 978-0-7695-1787-2. DOI: 10.1109/EURMIC.2002.1046193. URL: http://ieeexplore.ieee.org/document/1046193/ (visited on 10/15/2023).

Staszewski, W.J., K. Worden, and G.R. Tomlinson (Sept. 1997). “TIME–FREQUENCY ANALYSIS IN GEARBOX FAULT DETECTION USING THE WIGNER–VILLE DISTRIBUTION AND PATTERN RECOGNITION”. In: *Mechanical Systems and Signal Processing* 11.5, pp. 673–692. ISSN: 08883270. DOI: 10.1006/mssp.1997.0102. URL: https://linkinghub.elsevier.com/retrieve/pii/S0888327097901023 (visited on 11/25/2023).

Stol, Klaas-Jan, Paul Ralph, and Brian Fitzgerald (May 14, 2016). “Grounded Theory in Software Engineering Research: A Critical Review and Guidelines”. In: *Proceedings of the 38th International Conference on Software Engineering*. ICSE ’16: 38th International Conference on Software Engineering. Austin Texas: ACM, pp. 120–131. ISBN: 978-1-4503-3900-1. DOI: 10.1145/2884781.2884833. URL: https://dl.acm.org/doi/10.1145/2884781.2884833 (visited on 10/15/2023).

Stollwitzer, Andreas, Lara Bettinelli, and Josef Fink (Jan. 2023). “The Longitudinal Track-Bridge Interaction of Ballasted Track in Railway Bridges: Experimental Determination of Dynamic Stiffness and Damping Characteristics”. In: *Engineering Structures* 274, p. 115115. ISSN: 01410296. DOI: 10.1016/j.engstruct.2022.115115. URL: https://linkinghub.elsevier.com/retrieve/pii/S0141029622011919 (visited on 10/27/2023).

Swanson, Laura (Apr. 2001). “Linking Maintenance Strategies to Performance”. In: *International Journal of Production Economics* 70.3, pp. 237–244. ISSN: 09255273. DOI: 10.1016/S0925-5273(00)00067-0. URL: https://linkinghub.elsevier.com/retrieve/pii/S0925527300000670 (visited on 04/19/2024).

Symeonidis, Georgios et al. (Jan. 26, 2022). “MLOps - Definitions, Tools and Challenges”. In: *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*. 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC). Las Vegas, NV, USA: IEEE, pp. 0453–0460. ISBN: 978-1-66548-303-2. DOI: 10.1109/CCWC54503.2022.9720902. URL: https://ieeexplore.ieee.org/document/9720902/ (visited on 11/01/2023).

Szyperski, Clemens, Dominik W. Gruntz, and Stephan Murer (2009). *Component Software: Beyond Object-Oriented Programming*. 2. ed., [repr.] Addison-Wesley Component Software Series. London Munich: Addison-Wesley [u.a.] 589 pp. ISBN: 978-0-201-74572-6.

Tagliabue, Jacopo et al. (2023). “Reasonable Scale Machine Learning with Open-Source Metaflow”. Version 1. In: DOI: 10.48550/ARXIV.2303.11761. URL: https://arxiv.org/abs/2303.11761 (visited on 02/03/2024).

Teepe, Gerd and Thomas Görnig (2003). “Automotive Sensor Integration”. In: *Advanced Microsystems for Automotive Applications 2003*. Ed. by Jürgen Valldorf and Wolfgang Gessner. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 509–518. ISBN: 978-3-540-00597-1 978-3-540-76988-0. DOI: 10.1007/978-3-540-76988-0_40. URL: http://link.springer.com/10.1007/978-3-540-76988-0_40 (visited on 11/05/2023).

Tessaro, Iron, Viviana Cocco Mariani, and Leandro Dos Santos Coelho (Nov. 21, 2020). “Machine Learning Models Applied to Predictive Maintenance in Automotive Engine Components”. In: *The 1st International Electronic Conference on Actuator Technology: Materials, Devices and Applications.* The 1st International Electronic Conference on Actuator Technology: Materials, Devices and Applications. MDPI, p. 26. DOI: 10.3390/IeCAT2020-08508. URL: https://www.mdpi.com/2504-3900/64/1/26 (visited on 04/18/2024).

Testi, Matteo et al. (2022). “MLOps: A Taxonomy and a Methodology”. In: *IEEE Access* 10, pp. 63606–63618. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3181730. URL: https://ieeexplore.ieee.org/document/9792270/ (visited on 02/03/2024).

Theissler, Andreas et al. (Nov. 2021). “Predictive Maintenance Enabled by Machine Learning: Use Cases and Challenges in the Automotive Industry”. In: *Reliability Engineering & System Safety* 215, p. 107864. ISSN: 09518320. DOI: 10.1016/j.ress.2021.107864. URL: https://linkinghub.elsevier.com/retrieve/pii/S0951832021003835 (visited on 11/01/2023).

Tiwari, Umesh Kumar, Santosh Kumar, and Priya Matta (Oct. 2020). “Execution-History Based Reliability Estimation for Component-Based Software: Considering Reusability-Ratio and Interaction-Ratio”. In: *International Journal of System Assurance Engineering and Management* 11.5, pp. 1003–1019. ISSN: 0975-6809, 0976-4348. DOI: 10.1007/s13198-020-01035-1. URL: https://link.springer.com/10.1007/s13198-020-01035-1 (visited on 10/15/2023).

Ucar, Aysegul, Mehmet Karakose, and Necim Kırımça (Jan. 20, 2024). “Artificial Intelligence for Predictive Maintenance Applications: Key Components, Trustworthiness, and Future Trends”. In: *Applied Sciences* 14.2, p. 898. ISSN: 2076-3417. DOI: 10.3390/app14020898. URL: https://www.mdpi.com/2076-3417/14/2/898 (visited on 03/09/2024).

Urtado, Christelle, Huaxi Yulin Zhang, and Sylvain Vauttier (July 2010). “Architecture-Centric Development and Evolution Processes for Component-Based Software”. In: *Proceedings of 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE 2010).*

Utting, Mark and Bruno Legeard (2007). *Practical Model-Based Testing: A Tools Approach.* Amsterdam ; Boston: Morgan Kaufmann Publishers. 433 pp. ISBN: 978-0-12-372501-1.

Vaishnavi, Vijay K., Vijay K. Vaishnavi, and William Kuechler (May 6, 2015). *Design Science Research Methods and Patterns: Innovating Information and Communication Technology, 2nd Edition.* 0th ed. CRC Press. ISBN: 978-0-429-17220-5. DOI: 10.1201/b18448. URL: https://www.taylorfrancis.com/books/9781498715263 (visited on 10/15/2023).

Vallim Filho, Arnaldo Rabello De Aguiar et al. (May 19, 2022). “A Machine Learning Modeling Framework for Predictive Maintenance Based on Equipment Load Cycle: An Application in a Real World Case”. In: *Energies* 15.10, p. 3724. ISSN: 1996-1073. DOI: 10.3390/en15103724. URL: https://www.mdpi.com/1996-1073/15/10/3724 (visited on 11/01/2023).

Venable, John, Jan Pries-Heje, and Richard Baskerville (2012). “A Comprehensive Framework for Evaluation in Design Science Research”. In: *Design Science Research in Information Systems. Advances in Theory and Practice.* Ed. by Ken Peffers, Marcus Rothenberger, and Bill Kuechler. Red. by David Hutchison et al. Vol. 7286. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 423–438. ISBN: 978-3-642-29862-2 978-3-642-29863-9. DOI: 10.1007/978-3-642-29863-9_31. URL: http://link.springer.com/10.1007/978-3-642-29863-9_31 (visited on 10/15/2023).

Vollert, Simon, Martin Atzmueller, and Andreas Theissler (Sept. 7, 2021). “Interpretable Machine Learning: A Brief Survey from the Predictive Maintenance Perspective”. In: *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA).* 2021 IEEE 26th International Conference on Emerging Technologies and Factory Automation (ETFA). Vasteras, Sweden: IEEE, pp. 01–08. ISBN: 978-1-72812-989-1. DOI: 10.1109/ETFA45728.2021.9613467. URL: https://ieeexplore.ieee.org/document/9613467/ (visited on 11/01/2023).

Wallach, Hanna M. et al. (June 14, 2009). “Evaluation Methods for Topic Models”. In: *Proceedings of the 26th Annual International Conference on Machine Learning.* ICML ’09: The 26th Annual International Conference on Machine Learning Held in Conjunction with the 2007 International Conference on Inductive Logic Programming. Montreal Quebec Canada: ACM, pp. 1105–1112. ISBN: 978-1-60558-516-1. DOI: 10.1145/1553374.1553515. URL: https://dl.acm.org/doi/10.1145/1553374.1553515 (visited on 08/27/2023).

Waltman, Ludo, Nees Jan van Eck, and Anthony F. J. van Raan (Jan. 2012). “Universality of Citation Distributions Revisited”. In: *Journal of the American Society for Information Science and Technology* 63.1, pp. 72–77. ISSN: 15322882. DOI: 10.1002/asi.21671. URL: https://onlinelibrary.wiley.com/doi/10.1002/asi.21671 (visited on 08/25/2023).

Wang, Biao et al. (Aug. 2014). “Topic Selection in Latent Dirichlet Allocation”. In: *2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD).* 2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD). Xiamen, China: IEEE, pp. 756–760. ISBN: 978-1-4799-5148-2 978-1-4799-5147-5. DOI: 10.1109/FSKD.2014.6980931. URL: https://ieeexplore.ieee.org/document/6980931 (visited on 10/28/2023).

Wang, Jinjiang et al. (June 2017). “A New Paradigm of Cloud-Based Predictive Maintenance for Intelligent Manufacturing”. In: *Journal of Intelligent Manufacturing* 28.5, pp. 1125–1137. ISSN: 0956-5515, 1572-8145. DOI: 10.1007/s10845-015-1066-0. URL: http://link.springer.com/10.1007/s10845-015-1066-0 (visited on 11/25/2023).

Wang, Lintao et al. (June 1, 2022). “Fault Diagnosis and Predictive Maintenance for Hydraulic System Based on Digital Twin Model”. In: *AIP Advances* 12.6, p. 065213. ISSN: 2158-3226. DOI: 10.1063/5.0098632. URL: https://pubs.aip.org/adv/article/12/6/065213/2819337/Fault-diagnosis-and-predictive-maintenance-for (visited on 10/15/2023).

Wang, Tianyi et al. (Oct. 2008). “A Similarity-Based Prognostics Approach for Remaining Useful Life Estimation of Engineered Systems”. In: *2008 International Conference on Prognostics and Health Management.* 2008 International Conference on Prognostics and Health Management (PHM). Denver, CO, USA: IEEE, pp. 1–6. ISBN: 978-1-4244-1935-7. DOI: 10.1109/PHM.2008.4711421. URL: http://ieeexplore.ieee.org/document/4711421/ (visited on 08/25/2023).

Wang, Wenbin (Oct. 2012). “An Overview of the Recent Advances in Delay-Time-Based Maintenance Modelling”. In: *Reliability Engineering & System Safety* 106, pp. 165–178. ISSN: 09518320. DOI: 10.1016/j.ress.2012.04.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S0951832012000701 (visited on 11/25/2023).

Wang, Xiaolong et al. (Oct. 24, 2011). “Topic Sentiment Analysis in Twitter: A Graph-Based Hashtag Sentiment Classification Approach”. In: *Proceedings of the 20th ACM International Conference on Information and Knowledge Management.* CIKM ’11: International Conference on Information and Knowledge Management. Glasgow Scotland, UK: ACM, pp. 1031–1040. ISBN: 978-1-4503-0717-8. DOI: 10.1145/2063576.2063726. URL: https://dl.acm.org/doi/10.1145/2063576.2063726 (visited on 11/25/2023).

Wang, Yanlin et al. (2021). “CoCoSum: Contextual Code Summarization with Multi-Relational Graph Neural Network”. Version 1. In: DOI: 10.48550/ARXIV.2107.01933. URL: https://arxiv.org/abs/2107.01933 (visited on 02/25/2024).

Wang, Yiwei et al. (June 2017). “A Cost Driven Predictive Maintenance Policy for Structural Airframe Maintenance”. In: *Chinese Journal of Aeronautics* 30.3, pp. 1242–1257. ISSN: 10009361. DOI: 10.1016/j.cja.2017.02.005. URL: https://linkinghub.elsevier.com/retrieve/pii/S1000936117300201 (visited on 08/24/2023).

Washizaki, H., H. Yamamoto, and Y. Fukazawa (2003). “A Metrics Suite for Measuring Reusability of Software Components”. In: *Proceedings. 5th International Workshop on Enterprise Networking and Computing in Healthcare Industry (IEEE Cat. No.03EX717).* Ninth International Software Metrics Symposium. Sydney, NSW, Australia: IEEE Comput. Soc, pp. 211–223. ISBN: 978-0-7695-1987-6. DOI: 10.1109/METRIC.2003.1232469. URL: http://ieeexplore.ieee.org/document/1232469/ (visited on 01/16/2024).

Webster, Jane and Richard T. Watson (2002). “Analyzing the Past to Prepare for the Future: Writing a Literature Review”. In: *MIS Quarterly* 26.2, pp. xiii–xxiii. JSTOR: 4132319. URL: http://www.jstor.org/stable/4132319.

Wescoat, Ethan, Mihir Bangale, et al. (June 2023). “Physics Verification and Validation for Transferring Data between Bearings”. In: *Journal of Manufacturing Systems* 68, pp. 670–679. ISSN: 02786125. DOI: 10.1016/j.jmsy.2023.05.017. URL: https://linkinghub.elsevier.com/retrieve/pii/S0278612523000924 (visited on 08/25/2023).

Wescoat, Ethan, Matthew Krugh, Andrew Henderson, et al. (2019). “Vibration Analysis Utilizing Unsupervised Learning”. In: *Procedia Manufacturing* 34, pp. 876–884. ISSN: 23519789. DOI: 10.1016/j.promfg.2019.06.160. URL: https://linkinghub.elsevier.com/retrieve/pii/S2351978919308881 (visited on 08/25/2023).

Wescoat, Ethan, Matthew Krugh, Vinita Jansari, et al. (July 2023). “Redefining the Digital Triplet for Surrogate System Integration”. In: *Manufacturing Letters* 36, pp. 57–61. ISSN: 22138463. DOI: 10.1016/j.mfglet.2023.03.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S2213846323000159 (visited on 08/25/2023).

Wescoat, Ethan, Laine Mears, et al. (2020). “Frequency Energy Analysis in Detecting Rolling Bearing Faults”. In: *Procedia Manufacturing* 48, pp. 980–991. ISSN: 23519789. DOI: 10.1016/j.promfg.2020.05.137. URL: https://linkinghub.elsevier.com/retrieve/pii/S2351978920315894 (visited on 08/25/2023).

Wijayasiriwardhane, T., R. Lai, and K.C. Kang (2011). “Effort Estimation of Component-Based Software Development – a Survey”. In: *IET Software* 5.2, p. 216. ISSN: 17518806. DOI: 10.1049/iet-sen.2009.0051. URL: https://digital-library.theiet.org/content/journals/10.1049/iet-sen.2009.0051 (visited on 10/15/2023).

Will, R, M Mcquaig, and D Hardaway (Apr. 1994). “Identifying Long-Term Success Issues of Expert Systems”. In: *Expert Systems with Applications* 7.2, pp. 273–279. ISSN: 09574174. DOI: 10.1016/0957-4174(94)90043-4. URL: https://linkinghub.elsevier.com/retrieve/pii/0957417494900434 (visited on 11/05/2023).

Williams, L.G. (Nov. 15, 1991). *Formal Methods in the Development of Safety Critical Software Systems.* UCRL-ID–109416, 10146119, UCRL-ID–109416, 10146119. DOI: 10.2172/10146119. URL: http://www.osti.gov/servlets/purl/10146119-OoohxD/ (visited on 10/15/2023).

Wirth, Rüdiger and Jochen Hipp (2000). “CRISP-DM: Towards a Standard Process Model for Data Mining”. In: p. 11.

Wohlin, Claes (2012). *Experimentation in Software Engineering.* New York: Springer. ISBN: 978-3-642-29043-5.

Wolf, Yannic, Lennard Sielaff, and Dominik Lucke (2023). “A Standardized Description Model for Predictive Maintenance Use Cases”. In: *Procedia CIRP* 118, pp. 122–127. ISSN: 22128271. DOI: 10.1016/j.procir.2023.06.022. URL: https://linkinghub.elsevier.com/retrieve/pii/S2212827123002445 (visited on 11/26/2023).

Woodcock, Jim and Jim Davies (1996). *Using Z: Specification, Refinement, and Proof.* Prentice-Hall International Series in Computer Science. London ; New York: Prentice Hall. 386 pp. ISBN: 978-0-13-948472-8.

Woodcock, Jim, Peter Gorm Larsen, et al. (Oct. 2009). “Formal Methods: Practice and Experience”. In: *ACM Computing Surveys* 41.4, pp. 1–36. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/1592434.1592436. URL: https://dl.acm.org/doi/10.1145/1592434.1592436 (visited on 03/02/2024).

Wu, Dongyao et al. (Mar. 2016). “Building Pipelines for Heterogeneous Execution Environments for Big Data Processing”. In: *IEEE Software* 33.2, pp. 60–67. ISSN: 0740-7459. DOI: 10.1109/MS.2016.35. URL: http://ieeexplore.ieee.org/document/7420521/ (visited on 03/02/2024).

Wu, Ye, Dai Pan, and Mei-Hwa Chen (2001). “Techniques for Testing Component-Based Software”. In: *Proceedings Seventh IEEE International Conference on Engineering of Complex Computer Systems.* Seventh IEEE International Conference on Engineering of Complex Computer Systems. Skovde, Sweden: IEEE Comput. Soc, pp. 222–232. ISBN: 978-0-7695-1159-7. DOI: 10.1109/ICECCS.2001.930181. URL: http://ieeexplore.ieee.org/document/930181/ (visited on 10/15/2023).

Xia Cai et al. (2000). “Component-Based Software Engineering: Technologies, Development Frameworks, and Quality Assurance Schemes”. In: *Proceedings Seventh Asia-Pacific Software Engeering Conference. APSEC 2000.* Seventh Asia-Pacific Software Engineering Conference. ASPEC 2000. Singapore: IEEE Comput. Soc, pp. 372–379. ISBN: 978-0-7695-0915-0. DOI: 10.1109/APSEC.2000.896722. URL: http://ieeexplore.ieee.org/document/896722/ (visited on 10/15/2023).

Xiong, Deyi and Min Zhang (June 30, 2013). “A Topic-Based Coherence Model for Statistical Machine Translation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 27.1, pp. 977–983. ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v27i1.8566. URL: https://ojs.aaai.org/index.php/AAAI/article/view/8566 (visited on 08/27/2023).

Xiong, Deyi, Min Zhang, and Xing Wang (Mar. 2015). “Topic-Based Coherence Modeling for Statistical Machine Translation”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23.3, pp. 483–493. ISSN: 2329-9290, 2329-9304. DOI: 10.1109/TASLP.2015.2395254. URL: http://ieeexplore.ieee.org/document/7050388/ (visited on 10/28/2023).

Yildirim, Murat, Xu Andy Sun, and Nagi Z. Gebraeel (Nov. 2016). “Sensor-Driven Condition-Based Generator Maintenance Scheduling—Part I: Maintenance Problem”. In: *IEEE Transactions on Power Systems* 31.6, pp. 4253–4262. ISSN: 0885-8950, 1558-0679. DOI: 10.1109/TPWRS.2015.2506600. URL: http://ieeexplore.ieee.org/document/7387792/ (visited on 08/25/2023).

Yin, Robert K. (2018). *Case Study Research and Applications: Design and Methods.* Sixth edition. Thousand Oaks, California: SAGE Publications, Inc. ISBN: 978-1-5063-3615-2.

Zaharia, Matei et al. (2018). “Accelerating the Machine Learning Lifecycle with MLflow”. In: *IEEE Data Eng. Bull.* 41.4, pp. 39–45. URL: http://sites.computer.org/debull/A18dec/p39.pdf.

Zander, Justyna, Ina Schieferdecker, and Pieter J. Mosterman, eds. (2017). *Model-Based Testing for Embedded Systems.* 1st edition. Boca Raton: CRC Press. ISBN: 978-1-4398-1847-3.

Zhang, Xiaoling et al. (Oct. 17, 2021). “Transformer-XL With Graph Neural Network for Source Code Summarization”. In: *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC).* 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC). Melbourne, Australia: IEEE, pp. 3436–3441. ISBN: 978-1-66544-207-7. DOI: 10.1109/SMC52423.2021.9658619. URL: https://ieeexplore.ieee.org/document/9658619/ (visited on 02/25/2024).

Zhao, Rui et al. (Nov. 2016). “Machine Health Monitoring with LSTM Networks”. In: *2016 10th International Conference on Sensing Technology (ICST).* 2016 10th International Conference on Sensing Technology (ICST). Nanjing, China: IEEE, pp. 1–6. ISBN: 978-1-5090-0796-7.

DOI: 10.1109/ICSensT.2016.7796266. URL: http://ieeexplore.ieee.org/document/7796266/ (visited on 11/25/2023).

Zhao, Zhan, Haris N. Koutsopoulos, and Jinhua Zhao (Apr. 2018). "Individual Mobility Prediction Using Transit Smart Card Data". In: *Transportation Research Part C: Emerging Technologies* 89, pp. 19–34. ISSN: 0968090X. DOI: 10.1016/j.trc.2018.01.022. URL: https://linkinghub.elsevier.com/retrieve/pii/S0968090X18300676 (visited on 08/25/2023).

Zhou, Yue, Yue Yu, and Bo Ding (Oct. 2020). "Towards MLOps: A Case Study of ML Pipeline Platform". In: *2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*. 2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE). Beijing, China: IEEE, pp. 494–500. ISBN: 978-1-72819-146-1. DOI: 10.1109/ICAICE51518.2020.00102. URL: https://ieeexplore.ieee.org/document/9361315/ (visited on 11/01/2023).

Zio, Enrico and Michele Compare (Jan. 2013). "Evaluating Maintenance Policies by Quantitative Modeling and Analysis". In: *Reliability Engineering & System Safety* 109, pp. 53–65. ISSN: 09518320. DOI: 10.1016/j.ress.2012.08.002. URL: https://linkinghub.elsevier.com/retrieve/pii/S0951832012001524 (visited on 08/24/2023).

Zolnowski, Andreas and Tilo Böhmann (June 2014). "Formative Evaluation of Business Model Representations - The Service Business Model Canvas". In: ECIS 2014 Proceedings - 22nd European Conference on Information Systems.

Zonta, Tiago (2020). "Predictive Maintenance in the Industry 4.0: A Systematic Literature Review". In: *Industrial Engineering*, p. 17.