Titre: Title:	Smart Maintenance Framework Development Using Blockchain in an Industry 4.0 Context
Auteur: Author:	Salman Momen
Date:	2024
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Momen, S. (2024). Smart Maintenance Framework Development Using Blockchain in an Industry 4.0 Context [Master's thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/58313/

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Smart Maintenance Framework Development Using Blockchain in an Industry 4.0 Context

SALMAN MOMEN

Département de mathématiques et de génie industriel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

Génie industriel

Avril 2024

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Ce mémoire intitulé :

Smart Maintenance Framework Development Using Blockchain in an Industry 4.0 Context

présenté par **Salman MOMEN**

en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

a été dûment accepté par le jury constitué de :

Bruno AGARD, président

Christophe DANJOU, membre et directeur de recherche

Benoit EYNARD, membre

DEDICATION

I would like to express my deepest gratitude and heartfelt appreciation to my loving wife and incredible son for their unwavering support throughout the journey of writing my thesis. Your boundless patience, understanding, and encouragement have been the pillars that sustained me during the challenging times. Your belief in my abilities, the countless cups of coffee, and the late-night conversations filled with inspiration and motivation have truly been invaluable. Your unwavering love and constant reassurance propelled me forward when self-doubt threatened to overwhelm me. I am forever grateful for the sacrifices you both made, for creating a peaceful environment that allowed me to concentrate, and for always being there to celebrate each milestone along the way. This accomplishment is as much yours as it is mine. Thank you for being the driving force behind my success and for being my greatest cheerleader. I love you both more than words can express.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my supervisor and professor, Christophe Danjou, for his invaluable guidance, unwavering support, and expert knowledge throughout the entirety of my research. His insightful feedback, constructive criticism, and dedication to my academic growth have been instrumental in shaping this thesis. I am truly grateful for his mentorship and for instilling in me a passion for rigorous inquiry and intellectual curiosity.

I would also like to extend my gratitude to Payman Azizi, the production manager at Welbilt Company, for his collaboration and assistance during the case study phase of my research. Additionally, I extend my deepest appreciation to Professors Bruno Agard and Benoit Eynard for graciously accepting the responsibility of serving on my thesis jury. Their insightful comments, thoughtful suggestions, and meticulous evaluation have immensely contributed to the quality and rigor of this research. I am indebted to their expertise and critical analysis, which have helped shape and refine my arguments and conclusions.

Collectively, the contributions of Christophe Danjou, Payman Azizi, Professors Bruno Agard and Benoit Eynard have been invaluable to the completion of this thesis. Their support, guidance, and expertise have made this journey both challenging and rewarding. I am truly fortunate to have had their assistance and encouragement throughout this research endeavor.

RÉSUMÉ

L'avènement de l'Industrie 4.0 a révolutionné les opérations industrielles en introduisant des technologies avancées et des systèmes interconnectés. Dans ce contexte, la gestion efficace de la maintenance joue un rôle pivot pour garantir la performance optimale et la longévité des machines et équipements industriels. Cependant, les processus de maintenance traditionnels font face à des défis liés à la transparence, à la coordination et à la gestion des données. Cette mémoire présente un cadre novateur de maintenance intelligente conçu pour améliorer significativement l'efficacité des processus de maintenance, offrant ainsi une solution novatrice dans le contexte de l'Industrie 4.0.

L'objectif principal de cette recherche est d'étudier le potentiel de la technologie de la blockchain en tant que solution transformative pour améliorer la transparence, la traçabilité et l'efficacité dans la gestion de la maintenance. En exploitant la nature décentralisée et immuable de la blockchain, cette étude vise à rationaliser les opérations de maintenance, à promouvoir la responsabilité et à améliorer l'efficacité globale des processus. Une revue exhaustive de la littérature est menée pour analyser les travaux existants dans les domaines de l'Industrie 4.0, de la gestion de la maintenance et de la technologie de la blockchain, identifier les lacunes dans les connaissances et jeter les bases de la recherche.

S'appuyant sur les connaissances acquises de la revue de la littérature, une méthodologie est élaborée pour tirer parti de la technologie de la blockchain dans le processus de gestion de la maintenance. Cette méthodologie englobe des composants clés tels que la définition du processus d'existence, l'optimisation du processus par le biais de la cartographie, et la conception d'un cadre architectural exploitant les fonctionnalités inhérentes à la blockchain. De plus, une preuve de concept est mise en œuvre, impliquant l'utilisation de contrats intelligents pour automatiser et faire respecter les tâches et accords liés à la maintenance, ainsi qu'une interface utilisateur pour l'interaction des parties prenantes.

À travers cette recherche, des conseils précieux et des recommandations sont fournis aux praticiens et décideurs cherchant à optimiser les processus de maintenance et à exploiter pleinement le potentiel des avancées de l'Industrie 4.0, spécifiquement grâce à l'utilisation du cadre de la blockchain. En explorant les avantages de l'intégration de la blockchain, y compris une

transparence améliorée, une coordination améliorée et une gestion sécurisée des données, cette étude contribue au corpus existant des connaissances sur la gestion de la maintenance au sein de l'Industrie 4.0, offrant des implications pratiques pour la réalisation d'efficacités opérationnelles dans des environnements industriels.

ABSTRACT

The advent of Industry 4.0 has revolutionized industrial operations by introducing advanced technologies and interconnected systems. Within this context, effective maintenance management plays a pivotal role in ensuring the optimal performance and longevity of industrial machinery and equipment. However, traditional maintenance processes face challenges related to transparency, coordination, and data management. This dissertation introduces a pioneering smart maintenance framework designed to significantly enhance the proficiency of maintenance processes, offering a novel solution within the Industry 4.0 landscape.

The primary objective of this research is to investigate the potential of blockchain technology as a transformative solution for enhancing transparency, traceability, and efficiency in maintenance management. By leveraging the decentralized and immutable nature of blockchain, this study aims to streamline maintenance operations, foster accountability, and improve overall process efficiency. A comprehensive literature review is conducted to analyze existing works in the fields of Industry 4.0, maintenance management, and blockchain technology, identify gaps in knowledge, and lay the foundation for the research.

Building upon the insights gained from the literature review, a methodology is developed to leverage blockchain technology in the maintenance management process. This methodology encompasses key components such as defining the existence process, optimizing the process through mapping, and designing an architectural framework that leverages blockchain's inherent features. Additionally, a proof of concept is implemented, involving the use of smart contracts to automate and enforce maintenance-related tasks and agreements, along with a user interface for stakeholder interaction.

Through this research, valuable insights and recommendations are provided for practitioners and decision-makers seeking to optimize maintenance processes and harness the full potential of Industry 4.0 advancements, specifically through the utilization of the blockchain framework. By exploring the benefits of blockchain integration, including enhanced transparency, improved coordination, and secure data management, this study contributes to the existing body of knowledge on maintenance management within Industry 4.0, offering practical implications for realizing operational efficiencies in industrial settings.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

AI	Artificial Intelligence
ABI	Application Binary Interface
BC	Blockchain
BD	Big Data
BPM	Business Process Management
BPMN	Business Process Model and Notation
BPIIOT	A Light-Weighted Blockchain-Based Platform for Industrial IoT
CBM	Condition-based maintenance
CC	Cloud Computing
CPS	Cyber-Physical Systems
CPU	Central Processing Unit
Dapps	Decentralized Applications
DT	Digital Twin
ETH	Ether
EVM	Ethereum Virtual machine
FLIRT	Fast Light Innovative Regional Train
GRMI4.0	Guide for Representing and Modeling Industry 4.0
IDE	Integrated development environment
I4.0	Industry 4.0
IoT	Internet of Things
IIoT	Industrial Internet of Things
IT	Information Technology

M2M	Machine-to-Machine
ML	Machine Learning
OEE	Overall Equipment Effectiveness
PdM	Predictive Maintenance
PBFT	Practical Byzantine Fault Tolerance
PoA	Proof of Authority
PoB	Proof of Burn
PoC	Proof of Capacity
PoI	Proof of Importance
PoS	Proof of Stake
PoW	Proof of Work
RCM	Reliability-Centered Maintenance
RBM	Risk-based Maintenance
SOMD	Service-oriented maintenance decision-making
SDP	Software-defined perimeter
SDN	Software-defined Network
WDSF	Workflow-driven security framework
WoPeD	Workflow PetriNet Designer

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

In the dynamic landscape of Industry 4.0, where technology seamlessly intertwines with industrial processes, the role of maintenance management emerges as a critical factor for operational success. Across sectors, from manufacturing giants to small-scale enterprises, the efficient upkeep of machinery stands as the bedrock upon which productivity and profitability rest. However, amidst the digital crescendo of Industry 4.0, a significant portion of industrial maintenance remains entrenched in paper-based systems. This dissertation addresses the urgent question of how industries can seamlessly integrate their maintenance management into digital technology.

This exploration delves into the challenges faced by industries relying on manual, paper-bound maintenance tracking systems. The disparity between cutting-edge technology and archaic record-keeping methods raises concerns about accessibility, timely analysis, and the ability to implement proactive maintenance strategies. The lack of digitalization obstructs operational efficiency and hampers the ability to anticipate and prevent disruptions, highlighting the need for a transformative shift.

Within paper records lies the challenge of accountability and traceability. The absence of a digital trail diminishes the ability to assign responsibility, identify bottlenecks, and establish clear timelines for maintenance activities. The lack of traceability casts a shadow over industries, preventing them from achieving a holistic view of their maintenance practices. The urgency to bridge this digital gap and transform maintenance management practices is evident.

The central aim of this thesis is to elevate the efficiency and agility of maintenance processes while simultaneously enhancing transparency, traceability, and overall efficiency. In pursuit of this goal, the research introduces a smart maintenance framework tailored for maintenance management practices. Blockchain technology, renowned for its decentralized and immutable characteristics, emerges as a pivotal platform for ensuring secure and transparent data sharing and management. The utilization of blockchain's intrinsic features, including distributed consensus and data immutability, enables the streamlining of maintenance operations. Moreover, it fosters accountability and contributes to an overarching improvement in the efficiency of the maintenance management process.

To establish a foundation for the research, a comprehensive literature review is conducted, analyzing relevant works in the fields of Industry 4.0, maintenance management, and blockchain technology. Through a systematic examination of theoretical and practical articles, the current state of the field is assessed, and gaps in knowledge are identified. This literature review serves as a basis for developing a methodology that leverages blockchain technology in the maintenance management process.

The proposed methodology encompasses several key components, including defining the existence process, optimizing the process through mapping, and designing an architectural framework that leverages blockchain technology. Furthermore, a proof of concept is developed to validate the effectiveness of the proposed approach. This involves the implementation of smart contracts, which automate and enforce maintenance-related tasks and agreements, as well as a user interface that allows stakeholders to interact with the blockchain-based system.

The scientific contribution of this research lies in the comprehensive exploration and development of a smart maintenance framework within the context of Industry 4.0, leveraging the transformative capabilities of blockchain technology. The framework is meticulously designed to address the challenges inherent in analog maintenance tracking systems, offering a digitalized, decentralized, and immutable solution. By employing blockchain's unique features, such as distributed consensus and data immutability, the Smart Maintenance Framework aims to streamline maintenance operations, enhance transparency, and foster accountability. This novel approach not only contributes to the theoretical understanding of maintenance management but also provides a tangible and practical tool for industries navigating the complexities of Industry 4.0. The framework's potential to improve operational efficiency, communication effectiveness, and agility in maintenance processes positions it as a valuable asset for organizations seeking innovative solutions in the ever-evolving landscape of industrial maintenance.

The journey unfolds across six chapters, each addressing distinct facets of the research. Chapter 2 conducts a thorough literature review, exploring Industry 4.0, maintenance, and blockchain, identifying gaps, and paving the way for the research. Chapter 3 delineates research objectives, methodology, and the identified research gap. Chapter 4 presents the proposed process, outlining the existence process, technological integration, programming environment, smart contracts, and

the front-end interface. Chapter 5 focuses on the validation of the proposed framework, ensuring its efficacy in real-world.

CHAPTER 2 LITERATURE REVIEW

2.1 Industry 4.0

A comprehensive strategy leveraging connectivity and digitization by employing diverse technologies to revolutionize processes, products, and services. This approach emphasizes decentralized real-time decision-making, ushering in novel capabilities through collaborative efforts with humans. These capabilities span from system monitoring to control, optimization, and autonomy, creating a transformative paradigm in which technological advancements play a pivotal role (Danjou et al., 2017).

The term Industry 4.0, introduced in 2011, has become widespread in Germany, shaping discussions in various industry-related forums and sparking debates about its transformative nature. Industry 4.0's core technical foundation involves incorporating Internet technologies into industrial processes (Drath & Horch, 2014).

The advent of Industry 4.0, embracing concepts like digitalization, the Internet of Things (IoT), and Cyber-Physical Systems (CPS), marks a transformative shift in manufacturing. Originating from a German high-tech strategy, Industry 4.0 propels traditional companies into Smart Factories through the integration of IoT and CPS. The Industry 4.0 approach prioritizes decentralized management, fostering collaboration between the real and virtual realms within the network. This results in increased complexity and agility, promoting efficient data transmission and operational optimization across industries. The development of an IoT infrastructure further facilitates shared platforms via cloud systems in Supply Chains, ultimately streamlining business processes (Erboz, 2017).

In today's world, globalization and technological advancements have a profound impact on every aspect of our lives. We witness their influence in various domains, including shopping, entertainment, travel, design, communication, and manufacturing. Technology has seamlessly integrated into our daily routines, bringing about significant changes and advancements in information technology. In comparison to the past, our screen time has increased and our access to information has expanded. Technology now plays a crucial role in shaping our lives, driving notable transformations in how we interact with the world.

Before the 21st century, the world had witnessed three distinct industrial revolutions. The first began in the late 18th century, emphasizing the transition from hand production to mechanized manufacturing fueled by steam power. The second revolution emerged in the 19th century, particularly exemplified by the Ford Motor Company's adoption of continuous production methods and conveyor systems, leading to mass production capabilities. The third revolution commenced in 1969 with the introduction of the first programmable logic controller, leveraging the internet and electronic technologies to automate production processes. The ongoing fourth revolution, known as Industry 4.0, represents our current era of technological advancement (Drath & Horch, 2014; Lee et al., 2015). A comprehensive overview of these four industrial revolutions is depicted in figure 2.1.

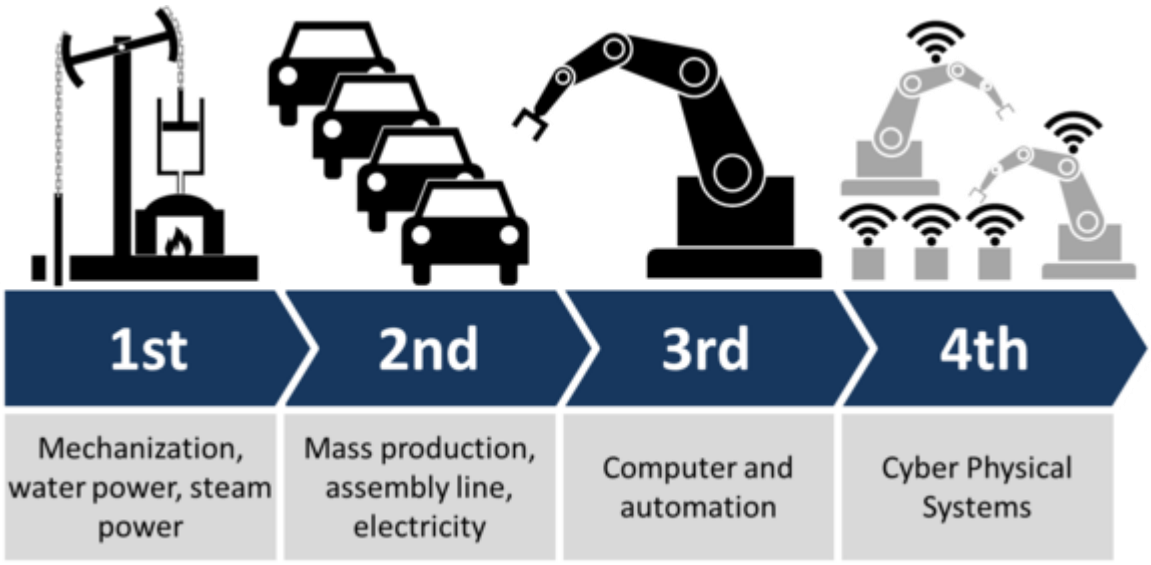


Figure 2.1 Overview of four industrial revolution (By Christoph Roser at AllAboutLean.com under the free CC-BY-SA 4.0 license)

From the first announcement of the fourth industrial revolution in 2011 to today we've observed the intensive growth of this revolution in different fields and theoretical concepts turn into the real applications of our daily life routines. People have started to adapt their life to these technologies and government policies encourage companies to invest in this context (Yang & Gu, 2021).

The anticipated fourth industrial revolution, Industry 4.0, is driven by key technologies, including cyber-physical systems (CPS), Internet of Things (IoT), Cloud Computing (CC), Big Data (BD),

Simulation, Digital Twins (DT), and Machine-to-Machine communication (M2M), as depicted in Figure 2.2. These enabling technologies adapt within Industry 4.0 to collect and communicate data, establishing a networked link among objects. The primary objective is to manage this data effectively, facilitating the reorganization and enhancement of production processes (Wichmann et al., 2019)

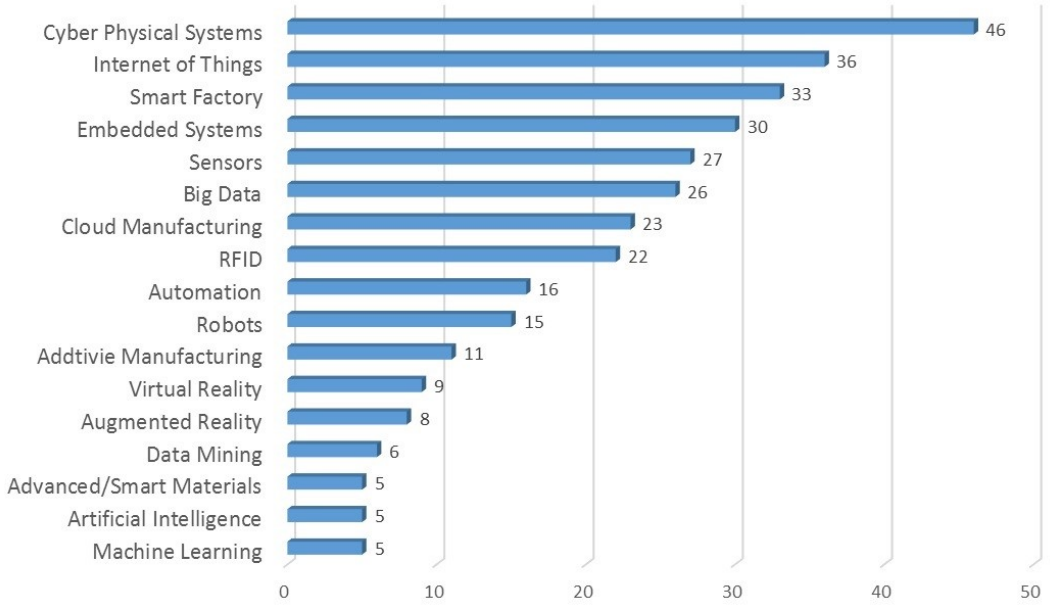


Figure 2.2 Key technologies instrumental to Industry 4.0 (Wichmann et al., 2019)

In the realm of Industry 4.0, characterized by its emphasis on advanced technologies and interconnected systems, this research uniquely contributes by using blockchain technology into maintenance management practices. Originating in 2011, Industry 4.0 prioritizes connectivity and digitization for transformative industrial processes.

2.2 Maintenance

Maintenance is defined by DIN 31051 as “combination of all technical, administrative and managerial actions during the life cycle of a facility or parts thereof, required to retain it in or restore it to a state in which it can perform the intended functions”(Stodt et al., 2019)

Maintenance comprises servicing, inspecting, overhauling, and improving. DIN 13306 classifies maintenance into two types: preventive maintenance and corrective maintenance. Preventive

maintenance aims to prevent machine failure, while corrective maintenance involves repairing a machine that has already failed. Preventive maintenance generally leads to higher savings than corrective maintenance. Moreover, a well-maintained machine translates to reduced downtime, thereby enhancing overall production (Stodt et al., 2019).

Qiuan et al. (2020) express that after the Second Industrial Revolution, the ability to design and manufacture products continually improved, and the technology to maintain equipment rapidly evolved, dividing maintenance into five stages. The first stage, occurring before 1950, is characterized as post-failure maintenance. The second stage, preventive maintenance, spans the period between 1950 and 1960. The third stage, predictive maintenance, was identified from 1960 until 1970. The fourth stage, parallel maintenance, extends from 1970 to 1995. The fifth stage, emerging after 1995 due to the rapid development and application of the internet and information technology, involves product maintenance based on the entire lifecycle (Qiuan et al., 2020).

Maintenance is a crucial aspect of ensuring the proper functioning of manufacturing systems. It is defined as the combination of all technical and administrative actions, including supervision, intended to maintain or restore an item to a state in which it can perform its required function. There are several different types of maintenance strategies, including reactive maintenance, preventive maintenance, predictive maintenance, and reliability-centered maintenance (RCM). (Moubray, 2001)

Reactive maintenance, also known as breakdown maintenance, is a strategy that involves fixing equipment after it has failed. Reactive maintenance is often the least costly option in the short term, as it requires minimal planning and no additional labor or materials (Patidar et al., 2017). However, reactive maintenance can lead to increased downtime and lost productivity, which can have a significant impact on overall equipment effectiveness (OEE) (Basri et al., 2017). In addition, reactive maintenance can lead to more significant and costly repairs in the long term, as well as potential safety risks (Patidar et al., 2017).

The hierarchical framework of maintenance types encompasses two main categories: planned maintenance and unplanned maintenance. Unplanned maintenance involves reactive measures taken unexpectedly to address failures, while planned maintenance includes preventive maintenance, predictive maintenance, Reliability Centered Maintenance (RCM), interval-based maintenance, and age-based maintenance. Preventive maintenance aims to maximize safety and

system reliability, while RCM optimizes efficiency, productivity, and cost by integrating various maintenance approaches, emphasizing unique schedules and tasks for individual equipment components. Age-based maintenance prevents component replacement prior to failure, while interval-based maintenance occurs at fixed time intervals, though it can be costly. Risk-based maintenance (RBM) focuses on maintaining critical systems to minimize failure likelihood, emphasizing the most risky machines. Condition-based maintenance (CBM) relies on sensor data to determine maintenance needs based on operational measurements like temperature and vibration. This framework provides diverse strategies for maintenance planning, catering to different aspects of system reliability and cost optimization (Patidar et al., 2017). The figure 2.3 shows the maintenance strategy categorizing.

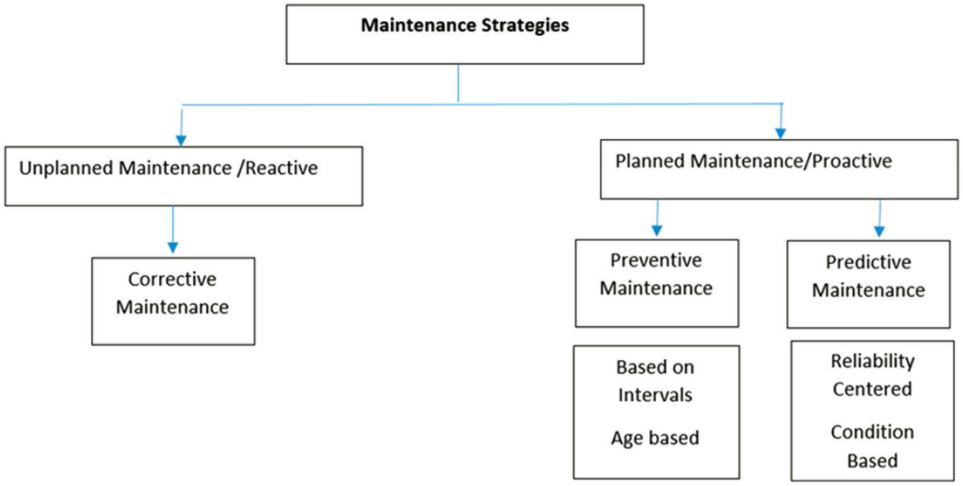


Figure 2.3 Maintenance Strategies (Patidar et al., 2017)

Preventive maintenance is a strategy that involves performing maintenance tasks on a regular schedule, regardless of whether the equipment has shown any signs of failure. Preventive maintenance is intended to prevent equipment failures before they occur, thereby reducing downtime and increasing OEE (Basri et al., 2017). Preventive maintenance can be time-consuming and costly in the short term, as it requires additional planning, labor, and materials (Patidar et al., 2017). Nevertheless, in the long term, preventive maintenance can result in cost reduction and increased productivity (Basri et al., 2017).

Predictive maintenance is a strategy that involves using data and analytics to predict when equipment is likely to fail, allowing maintenance to be performed before failure occurs (Patidar et al., 2017). Predictive maintenance can be more effective than preventive maintenance, as it allows

maintenance to be performed only when necessary, reducing downtime and increasing OEE (Basri et al., 2017). However, predictive maintenance requires significant investment in data analysis and equipment monitoring systems, making it more costly than other maintenance strategies (Patidar et al., 2017).

Reliability-centered maintenance (RCM) is a strategy that involves identifying the most critical components of a manufacturing system and developing maintenance strategies specifically tailored to those components (Moubray, 2001). RCM takes into account the consequences of failure, the likelihood of failure, and the cost of maintenance to develop a maintenance plan that is both effective and efficient (Basri et al., 2017). Despite the potential for significant cost savings and increased OEE, RCM demands extensive planning and analysis (Patidar et al., 2017).

Industry 4.0 has brought about a significant shift in the way manufacturing processes are conducted, with a focus on collaboration across different organisations and leveraging smart technologies to optimise these processes. One key area where this has had a transformative impact is maintenance. With the Internet of Things and smart sensors, it has become possible to collect a vast amount of data about the performance of machines and equipment in real-time. By applying advanced data analytics and cloud computing, this data can be used to enable predictive maintenance, which allows for the early detection, location, and diagnosis of potential faults. This not only minimises downtime and reduces costs but also ensures that machines are operating at their optimal level (Sang et al., 2021).

Due to the rising complexity of interactions between different production activities in a constantly broad manufacturing environment, maintenance has become critical for industries. Industrial internet of things (IIoT) Machine Learning (ML) and Big Data (BD) add value to the predictive maintenance (PdM) (Zonta et al., 2020).

Within the realm of maintenance, this research underscores corrective maintenance, specifically addressing machinery repairs after failure as per DIN 13306. The investigation meticulously optimizes post-failure scenarios, enhancing the efficiency of corrective maintenance processes. Importantly, the research framework holds potential for broader applications, with adaptable methodologies extending beyond corrective maintenance to include preventive strategies and other maintenance facets.

2.3 Blockchain

Blockchain is a decentralized digital ledger technology that has transformed various sectors, including businesses and commerce, by eliminating the requirement for a central storage and control authority. It introduces timestamped and unalterable data blocks that are not under the ownership of a single entity. Instead, they are managed by a network of nodes or computers, with each block secured and interconnected through cryptographic principles. The unalterable and decentralized characteristics of blockchain have reshaped concepts of trust, ownership, identity, and financial systems, offering a secure, rapid, transparent, and partially anonymous solution (Tripathi et al., 2023).

In 2009, Satoshi Nakamoto introduced blockchain technology through his paper "Bitcoin: A Peer-to-Peer Electronic Cash System," establishing the mathematical foundation for the cryptocurrency Bitcoin. This groundbreaking system proposes a trustless approach to electronic transactions, leveraging digital signatures and a peer-to-peer network with proof-of-work to prevent double-spending. The unstructured simplicity of the network allows nodes to work independently, requiring minimal coordination, and enables them to freely join and leave while expressing their acceptance or rejection of blocks through CPU power, creating a robust consensus mechanism (Nakamoto, 2009).

A blockchain functions as a sequential chain of blocks, serving as a comprehensive ledger for transaction records. The initial block, known as the genesis block, lacks a parent block. The core components of a block, such as the block version, Merkle tree root hash representing transaction data, timestamp indicating the current universal time, nBits specifying the target threshold for a valid block hash, nonce for hash calculation, and the parent block hash linking to the preceding block, are elucidated the block structure shown in Figure 2.4. Additionally, users possess a pair of private and public keys, with the private key used for transaction signing, a process involving two stages: signing and subsequent verification. These digitally signed transactions are then disseminated across the network (Zheng et al., 2017).

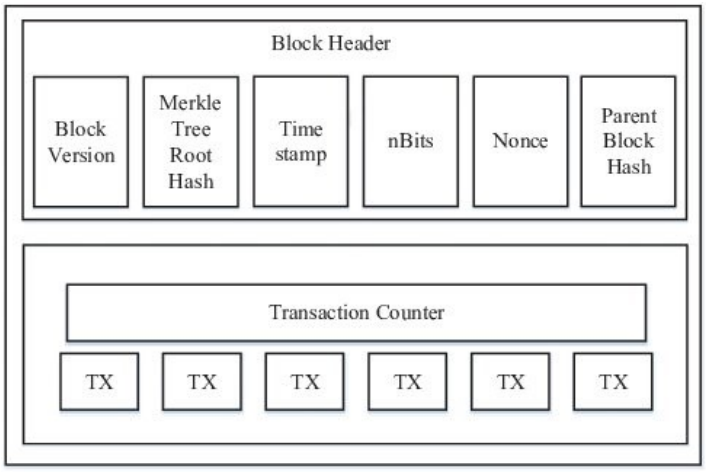


Figure 2.4 Block Structure (Zheng et al., 2017)

In the blockchain, blocks are linked together through reference to the hash value of the previous block. Each node within the network possesses the entire chain and appends a new block whenever necessary. Upon insertion of a new block into the chain, its index must surpass that of the latest block, and the block hash must meet the specified difficulty requirement. Verification of each node precedes its addition to the network. The computed hash value results from combining the hash of the previous block with the block's own data, incorporating all items, and including the timestamp. This process facilitates the creation of a structured chain of interconnected blocks. The addition of a block is contingent upon the matching hash value of the preceding block and the unanimous approval of all network members the chain formation shown in Figure 2.5 (Komalavalli et al., 2020).

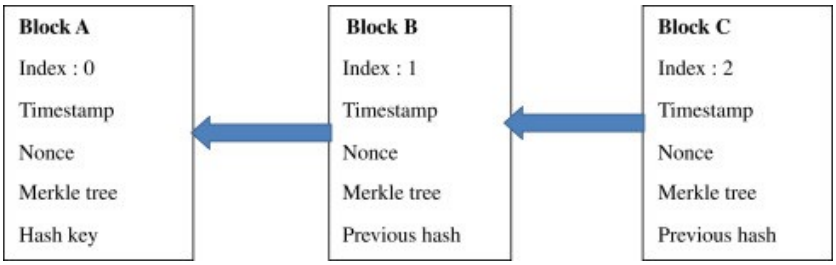


Figure 2.5 Chain formation (Komalavalli et al., 2020)

Consensus algorithms play a crucial role in determining how blocks are incorporated into the blockchain structure. In a blockchain network, a shared agreement is reached among all nodes to facilitate the addition of a new block, ensuring trust among unfamiliar peers within the network (Komalavalli et al., 2020).

The first algorithm, Proof of Work (PoW), originated with Bitcoin and is extensively employed in the mining process. Miners engage in solving a cryptographic puzzle to find the hash value for the next block. The network nodes then reach a consensus on the miner's hash, allowing the block to be added. The initial miner solving the puzzle receives a reward, but the process demands substantial computational power. Another algorithm, Proof of Stake (PoS), validates blocks based on participants' stakes, with validation determined by the currency invested. Miners, however, do not receive monetary rewards. Practical Byzantine Fault Tolerance (PBFT) addresses node failures within the network, focusing on both faulty and operational nodes. Achieving fault tolerance involves considering correct values from operational nodes while assigning default vote values for faulty nodes, leading to a network consensus on accurate values (Bhushan et al., 2021; Komalavalli et al., 2020; Tripathi et al., 2023).

Proof of Activity (PoA) combines PoW and PoS, assuring the authenticity of all transactions. In the first phase, miners use the PoW consensus to identify a new block, after which the process transitions to PoS. Proof of Burn Time (PoB) involves burning or destroying coins held by miners, ensuring network agreement on the valid state, and preventing double-spending. This algorithm addresses the energy consumption associated with PoW by permitting miners to write a block proportionate to the number of tokens destroyed. Proof of Capacity (PoC) utilizes hard disk space instead of computational power, allowing miners to store potential solutions on the hard drive before mining. This minimizes changes in the header value and enables miners to match the required hash value from a list to win rewards. Finally, Proof of Importance (PoI), based on PoS, operates on the concepts of harvesting and vesting. Harvesting determines a node's eligibility to add a block, and the node vests transaction fees within that block (Bhushan et al., 2021; Komalavalli et al., 2020; Tripathi et al., 2023).

Blockchain architecture relies on two fundamental aspects: the ownership of the data infrastructure, determined by whether the blockchain is public or private, and the permissions associated with various operations like joining, reading, writing, and committing. The architectural solutions are developed based on these considerations, leading to the categorization of open and closed blockchains according to write permissions and public and private blockchains according to read permissions. Public and private blockchains are often defined in terms of who can write data, resulting in categories such as open public, closed public, open private, and closed private blockchains. You can see the blockchain types in Figure 2.6 (Tripathi et al., 2023).

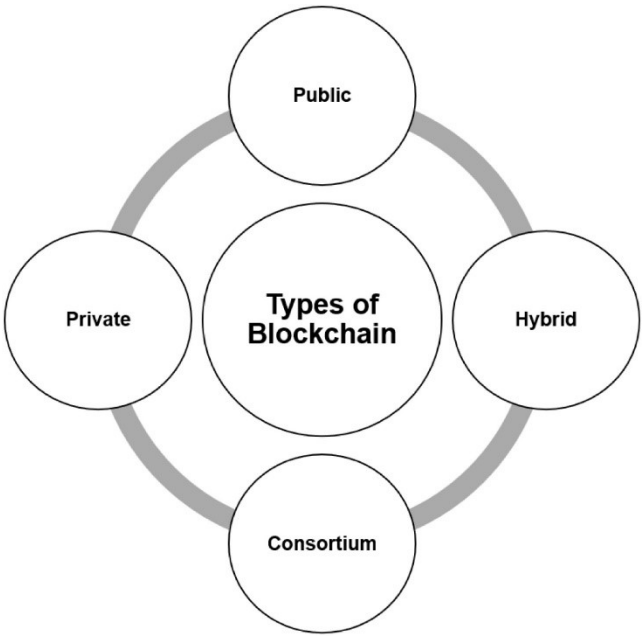


Figure 2.6 Types of blockchain (Tripathi et al., 2023)

In the context of blockchain types, which are characterized by consensus, distributed computation, immutability, and authentication, the distinction is made based on the domain and users of the application, with user authentication playing a crucial role. Public chains, characterized as open-source and permissionless ledgers, allow any participant to join without requiring permission. Examples include Bitcoin and Ethereum. On the other hand, private chains are permissioned ledgers that restrict access to selected participants, ensuring that only authorized users within a specific organization control and interpret encrypted blocks. Examples of private blockchains are Hyperledger Fabric, Corda, and Quorum. Hybrid blockchains, also known as consortium blockchains, combine features of both public and private blockchains, forming a semi-decentralized and semi-private structure. This type supports controlled user groups while functioning across different organizations (Komalavalli et al., 2020).

A smart contract functions as a computer code-driven agreement executed without reliance on third parties, ensuring trustless transactions. This innovation extends the reach of blockchain technology beyond cryptocurrencies, finding applications in diverse fields such as supply chain, IoT, healthcare, and business process management. With similar security characteristics to blockchain data, smart contracts specify trigger conditions, participant obligations, rights, and results. The accuracy and objectivity of blockchain execution hinge on the secure implementation logic of smart

contract code, emphasizing the critical role of thorough testing and professional review for verification purposes (Bhushan et al., 2021).

Blockchain's rapid growth introduces a set of challenges that require simultaneous attention alongside its advancements. A notable concern is scalability, particularly in the banking sector, where blockchain struggles to handle the sheer volume of daily financial transactions. The inability to directly modify data in blockchain systems necessitates creating new nodes for modifications, resulting in decreased transaction speed and increased fees. While scalability issues could be addressed through storage optimization and reevaluating block size, finding the right trade-off remains complex (Bhushan et al., 2021; Komalavalli et al., 2020; Tripathi et al., 2023)

Another cluster of challenges includes public perception, cultural shift, initial costs, integration with legacy systems, security, privacy, legal regulations, energy consumption, and a shortage of skilled personnel. Public perception suffers from a lack of knowledge beyond Bitcoin, hindering exploration in other fields. The cultural shift toward decentralized digital technologies raises concerns about trust and security. The initial cost of blockchain adoption, especially for banks, poses a hurdle. Integrating blockchain with legacy systems demands careful planning, and security and privacy concerns necessitate private blockchain implementations. Legal regulations become crucial, and energy consumption, particularly with Proof-of-Work, raises environmental concerns. Shortages in skilled personnel compound these challenges, emphasizing the need for a comprehensive strategic approach to unlock blockchain's full potential (Komalavalli et al., 2020; Tripathi et al., 2023)

2.4 **Systematic literature review**

Although, blockchain has introduced to the world more than a decade ago but focusing on manufacturing and maintenance in blockchain has not attracted the attention of researchers. Looking for articles in the English language on the Web of Science and engineering village website drives us to find articles relevant to the research topics and provides us access to scientific issues related to our thesis. Using these two reference webs following results have been found.

2.4.1 Keywords and query

Defining precise keywords is fundamental in constructing a comprehensive literature review. These keywords serve as the cornerstone for identifying relevant academic resources, guiding the focus of the research. In our case, the process of determining these keywords begins by delineating the scope of our work. This involves careful consideration of the technologies central to our study, encompassing blockchain and its variants like "block chain". Additionally, the industry sector is crucial, leading us to include terms such as production, manufacturing, and industrial environment. Lastly, our thesis delves into the domain of services utilizing this technology, notably maintenance. Categorizing these keywords aligns our research with the core elements of technology, industry, and services, ensuring a targeted and effective literature search.

In the realm of technology, our focus narrows down to blockchain, a revolutionary decentralized ledger system transforming various sectors. The term "blockchain" and its alternative spelling "block chain" are pivotal keywords. These encompass the underlying technology that our research revolves around. Blockchain, with its cryptographic security and decentralized nature, is a linchpin in our investigation. By exploring both spellings, we broaden our search scope, ensuring inclusivity and meticulousness in our literature review. This technological foundation is pivotal in understanding the intricacies of its integration within industrial frameworks.

Contextualizing our research within the industrial landscape is imperative. We incorporate keywords such as production, manufacturing, and industrial environment to anchor our study within these sectors. Production and manufacturing highlight the core processes our thesis addresses, encompassing the creation and assembly of goods. Simultaneously, the term industrial environment encompasses the broader setting, integrating various industrial facets where blockchain technology finds application. These keywords illuminate our exploration of how blockchain augments and optimizes processes within these industrial domains. Through these carefully selected terms, our literature review will delve into the heart of industrial applications, providing a nuanced perspective essential to our research objectives.

In the realm of services, our thesis zeroes in on maintenance, a critical aspect in the industrial landscape. Maintenance, the systematic process of preserving, repairing, and optimizing machinery and systems, is central to our study. It ensures the seamless operation of industrial processes and aligns with our overarching theme. By incorporating the keyword "maintenance," our literature

review delves into the intricate fusion of blockchain technology within maintenance protocols. This keyword acts as a guiding beacon, illuminating research articles and scholarly works exploring the symbiotic relationship between blockchain and maintenance services. Through this exploration, our thesis aims to unearth innovative applications, challenges, and solutions, contributing valuable insights to the intersection of blockchain technology and maintenance practices in industrial contexts. We consider the most important of these keywords as you can see in Table 2-1.

Table 2-1Keywords

Industry	Technologies	Services
Production	Blockchain	Maintenance
Manufacturing	Block chain	
Industrial Environment		

The Searching query was the following:

(("Blockchain" OR "block chain") AND (Maintenance) AND (manufacturing OR production or "industrial environment"))

2.4.2 Search methodology and results

The first literature review was completed in October 2021, and the final review was completed in November 2022, and updated in August 2023, yielding 189 articles in engineering village and web of science. After reading the abstracts, we found that 8 of the publications' abstracts were related to our issue. Following that, a thorough reading of each item was initiated, 6 articles to be theoretical, and only 2 articles to be practical. Figure 2.7 depicts the flow of the literature review.

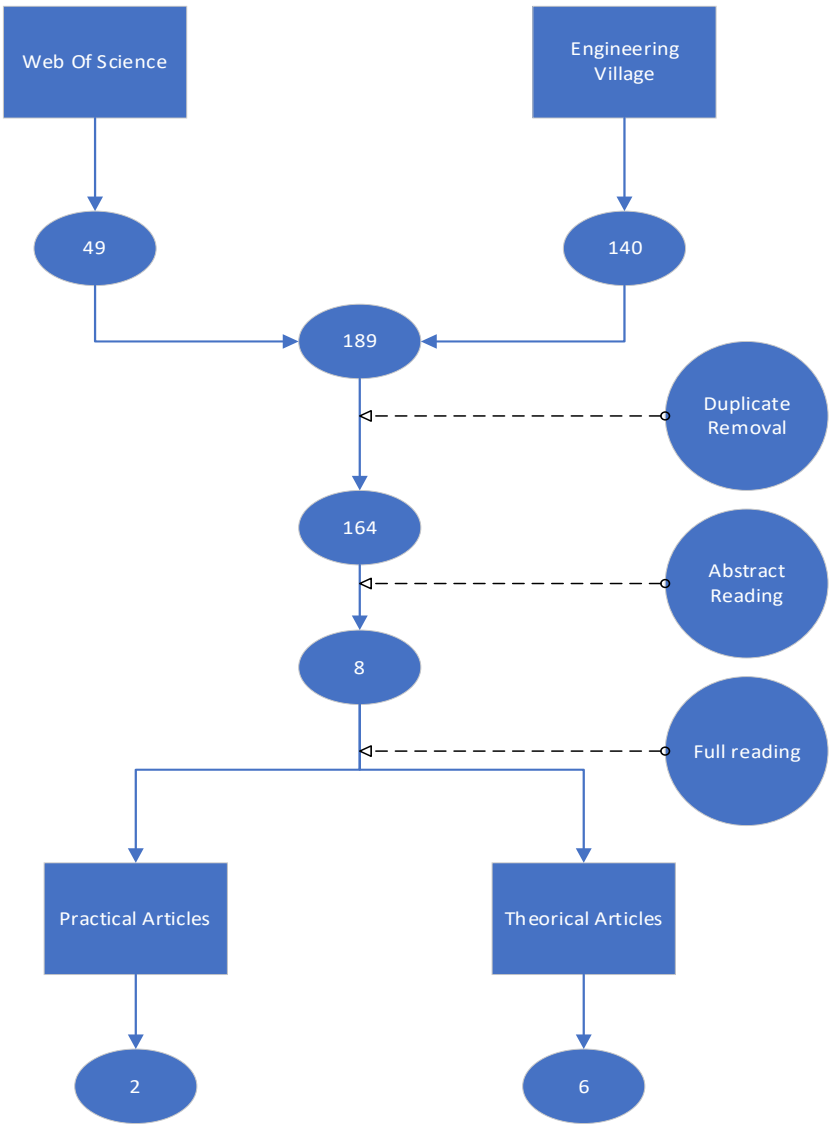


Figure 2.7 Literature review flow

As mentioned above after reading the abstract of the selected articles only 8 articles seem related to our topic. After fully reading of theses articles two types of articles identified,6 articles are theoretical work, and 2 articles are practical work. In following this analyzed has been discussed.

2.4.2.1 Theoretical articles

Theoretical articles constitute the bedrock of academic research, providing foundational concepts, frameworks, and hypotheses upon which practical applications are built. In our literature review, theoretical articles form a pivotal category. These scholarly works delve into the fundamental principles of blockchain technology, exploring its cryptographic algorithms, consensus

mechanisms, and decentralized nature. They offer theoretical foundations on which our research is anchored, shedding light on the underlying mechanisms of blockchain networks. Furthermore, theoretical articles often explore conceptual models and propose hypotheses, providing valuable insights into the potential applications of blockchain in industrial maintenance. By critically analyzing these theoretical contributions, our literature review establishes a strong theoretical framework, enabling a deeper understanding of the theoretical underpinnings of blockchain technology and its integration within maintenance practices.

Wallis in "Agreements between Enterprises digitized by Smart Contracts in the Domain of Industry 4.0" studied interconnection between different companies and information sharing various stakeholders and Blockchain's advantages in comparison with central database and trusted third-party. The paper shows how different agreements use maintenance use cases to show the benefits of digital contracts and the challenges and solutions for digital contracts. In the maintenance, case use the possible error occurrence in traditional paper-based and smart contracts identified (Wallis et al., 2020).

Welte et al (2020) proposed a blockchain centric architecture of integration of blockchain into manufacturing, focusing on, but not limited to, maintenance for secure connection of data sources such as software and devices to blockchain in an industrial environment and mentioned that it is necessary to adopt the blockchain to them because of heterogeneity of the system to be connected. In the paper they identify potential threats and potential countermeasures. We can see this architecture in Figure 2.8 (Welte et al., 2020).

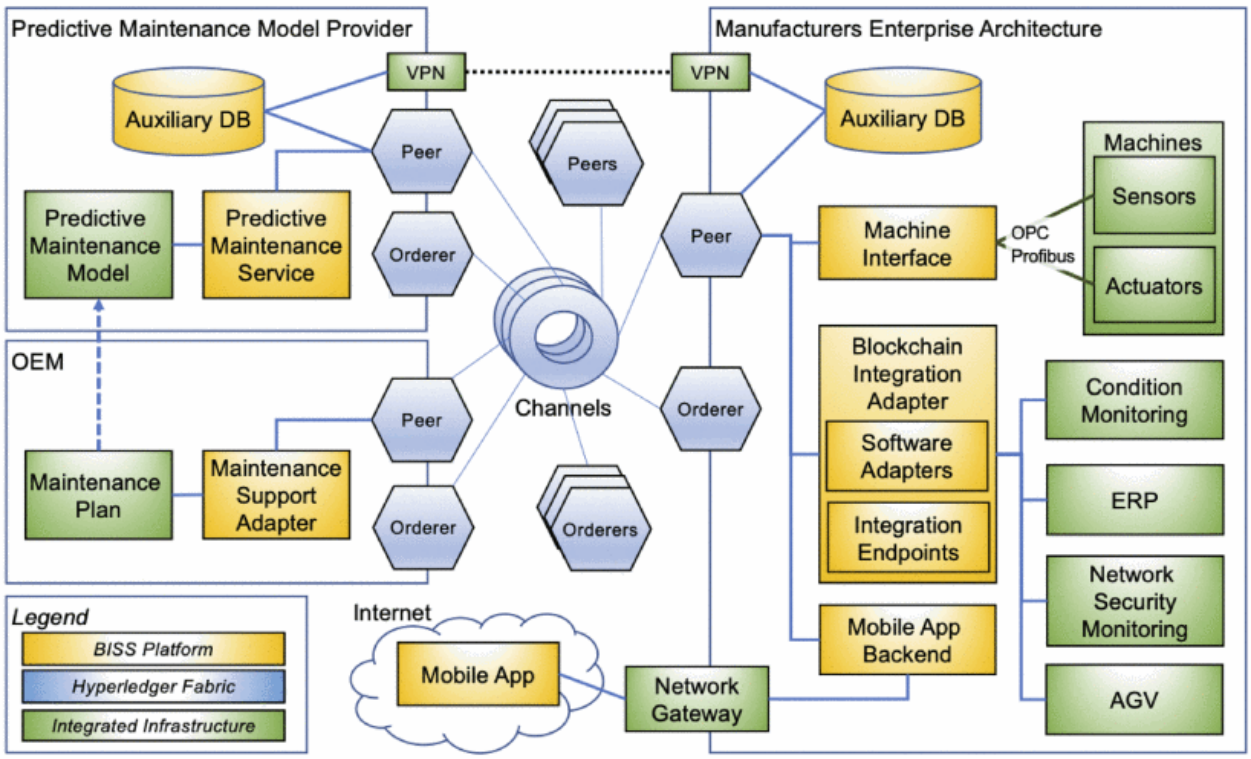


Figure 2.8 Architecture of interconnected system to centric blockchain (Welte et al., 2020)

Chang et al. (2021) provide a network architecture for knowledge sharing with a focus on service-oriented maintenance decision-making (SOMD) to collect, manage, and share obtained resources between different stakeholders involved in maintenance performing (Chang et al., 2021).

The authors in the “Intelligent Maintenance of Complex Equipment Based on Blockchain and Digital Twin Technologies“ paper propose a framework based on digital twin and blockchain technologies for complex equipment which consist of four modules, which are complex equipment as a physical entity, Blockchain as a distributed data system, digital twin for integrating physical entities and digital entities, and intelligent maintenance which is the application of digital twin and block chain in order to realize condition monitoring, fault diagnosis and life prediction of complex equipment. The overall framework shown in the figure 2.9 (Qiuan et al., 2020).

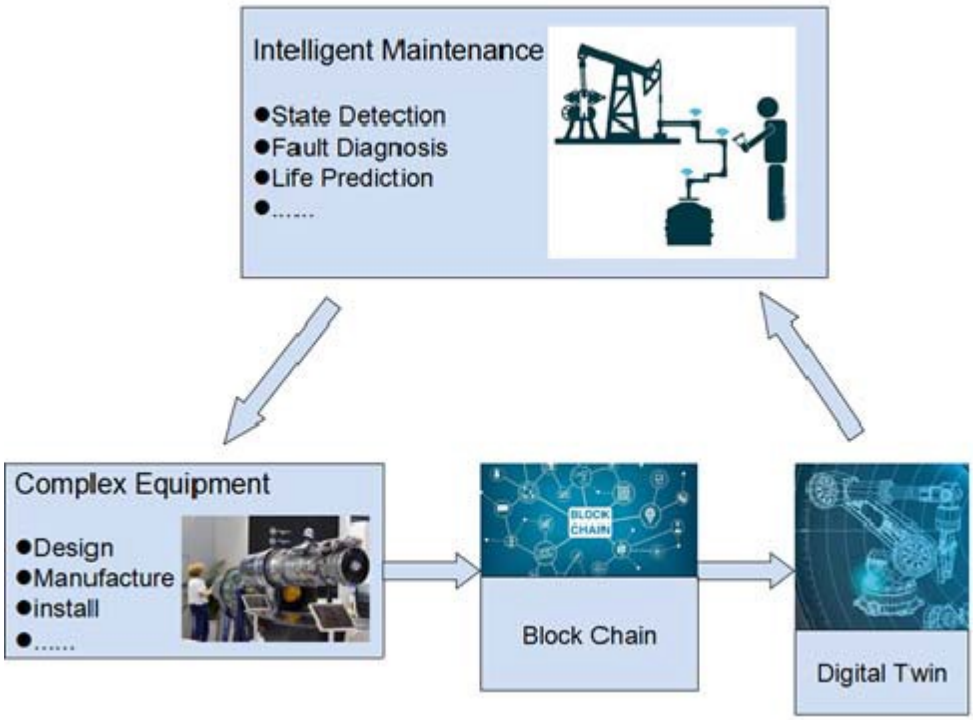


Figure 2.9 The overall framework (Qiuan et al., 2020)

Li et al. (2019) explains the concept definition and technical principle and infrastructure of blockchain and propose alight weighted platform so-called BPIIoT which is based on a blockchain network embedded with smart contracts which is consist of on-chain network and off-chain network to increase the speed of the network and provide a secure platform for sharing information (Li et al., 2019).

Stodt et al. (2019) proposed a concept of integrating maintenance processes utilizing blockchain for achieving a higher degree of transparency and enhancing trust between stakeholders. This concept has forced the stakeholders to perform their tasks in better quality and missed maintenance and poorly performed maintenance could be identified (Stodt et al., 2019).

2.4.2.2 Practical articles

Practical articles, the heart of our literature review, delve into real-world applications and case studies where blockchain technology is implemented in industrial maintenance contexts. These

articles provide invaluable insights into the challenges faced by industries, the innovative solutions devised, and the measurable impacts of blockchain integration. Practical articles showcase diverse industrial sectors, from manufacturing plants to supply chain management, where blockchain has been effectively utilized in maintenance processes. By scrutinizing these practical applications, our literature review captures the pulse of industry trends, highlighting successful implementations and identifying areas of improvement. These articles serve as a bridge between theory and real-world application, offering tangible examples and empirical evidence that substantiate our research objectives. Through the analysis of practical articles, our literature review synthesizes existing knowledge, paving the way for a nuanced understanding of the practical implications of integrating blockchain technology in industrial maintenance practices.

In "Secure Remote Maintenance via Workflow-Driven security Framework" Requirements of industrial remote maintenance for security protection have been identified and analyzed and present a secure maintenance case scenario in an industrial environment to show how those protection requirements could be addressed by using a workflow-driven security framework (WDSF). To mitigate the threats and perform the workflows The WDSF is combined with state-of-the-art technologies such as blockchain, smart contracts, and SDP/SDN firewalls. An architecture of the WDSF has been presented and different layers were identified. The workflow processes were modeled by Petri Net and validation has been done by WoPeD (workflow PetriNet Designer). The WDSF deployment architecture shows in Figure 2.10 (Kasinathan et al., 2021).

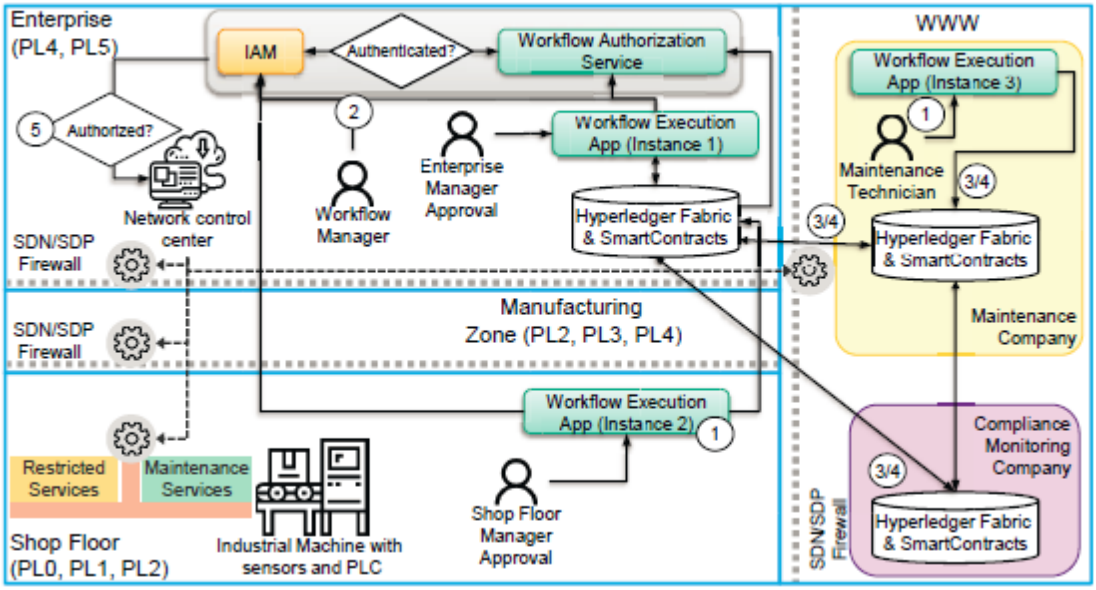


Figure 2.10 WDSF Deployment Architecture (Kasinathan et al., 2021)

Abbas et al. (2020) develop an application to enhance the trust between stakeholders at the Fast Light Innovative Regional Train (FLIRT) in Netherlands. They use Hyperledger Fabric for the Blockchain architecture and the application user interact with application written in Angular 4 based on defined protocol. Figure 2.11 show the information architecture (Abbas et al., 2020).

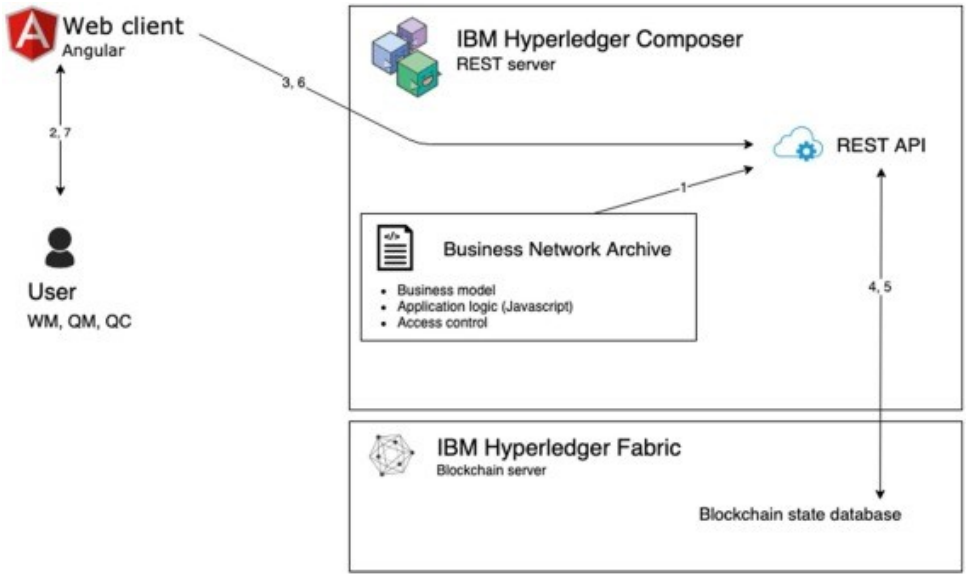


Figure 2.11 Information architecture of the developed application

2.5 Conclusion

In conducting a comprehensive literature review, a critical analysis has revealed both positive and negative aspects inherent in the selected articles. The identification and categorization of articles into relevant and non-relevant groups have been instrumental in refining the focus of our research on blockchain technology in industrial maintenance.

Non-relevant articles, despite their academic merit, have been recognized for their divergence from our specific research focus. These articles span a wide range of topics within the broader field of blockchain, including cryptocurrency applications and social implications. Acknowledging the intellectual stimulation they offer; we have judiciously categorized them as non-relevant to our study. This strategic discernment has enabled us to streamline our literature review, ensuring a meticulous and focused analysis of articles directly aligned with our research objectives.

Theoretical articles, constituting a fundamental category, serve as the intellectual foundation of our research. These scholarly works delve into the core principles of blockchain technology, exploring cryptographic algorithms, consensus mechanisms, and the decentralized nature of blockchain networks. By critically analyzing these theoretical contributions, our literature review establishes

a robust theoretical framework, facilitating a deeper understanding of the theoretical underpinnings of blockchain technology and its potential integration within maintenance practices.

Practical articles, at the heart of our literature review, offer insights into real-world applications and case studies where blockchain technology is implemented in industrial maintenance contexts. These articles provide a bridge between theory and practice, showcasing diverse industrial sectors and highlighting successful implementations. Through the scrutiny of practical applications, our literature review synthesizes existing knowledge, paving the way for a nuanced understanding of the practical implications of integrating blockchain technology in industrial maintenance practices.

CHAPTER 3 RESEARCH OBJECTIVES AND METHODOLOGY

3.1 Research gap

This comprehensive approach, encompassing theoretical foundations and practical applications, ensures a well-rounded analysis that enhances the depth and relevance of our research findings. In the landscape of blockchain applications in the context of Industry 4.0, a critical examination of the existing literature reveals a discernible research gap in the specific area of smart maintenance. While numerous studies explore the broader integration of blockchain technology in industrial settings, a focused examination of smart maintenance, facilitated through the development of a dedicated smart contract framework, remains notably scarce. The available literature often touches upon the potential of blockchain in enhancing industrial processes, including maintenance, but lacks a comprehensive investigation into the intricacies of developing a dedicated smart contract framework tailored for smart maintenance practices. This research gap underscores the need for a specialized exploration that not only addresses the theoretical foundations of blockchain in Industry 4.0 but also delves into the practical implementation aspects required for effective and efficient smart maintenance.

Furthermore, existing studies often provide a theoretical understanding of blockchain technology in an industrial context but fall short in offering a holistic framework specifically designed for smart maintenance. The literature predominantly focuses on the theoretical underpinnings of blockchain, its cryptographic principles, and its decentralized nature, leaving a void in the literature concerning the detailed development of a smart contract framework for smart maintenance. Consequently, this research gap necessitates a dedicated investigation to bridge the divide between theoretical concepts and practical implementation, ultimately contributing to the advancement of Industry 4.0 by fostering innovation in maintenance practices through the integration of blockchain technology.

3.2 Research objective

The objective of this thesis is to increase the efficiency and agility of the maintenance process and improving transparency, traceability, and efficiency. To reach this aim we propose and develop an innovative smart maintenance framework, BlockMaint4.0, featuring a novel process seamlessly

aligned with Industry 4.0 principles. This forward-looking framework, harnessing the power of blockchain technology, seeks to establish a robust and secure foundation for the execution of smart maintenance processes in industrial settings. The focus is on suggesting a new process that not only incorporates theoretical insights from blockchain literature but also addresses the practical requirements specific to industrial maintenance within the Industry 4.0 context.

Research question 1: What defines an agile and smart maintenance process, and do our current maintenance practices embody these characteristics? This research question is dedicated to evaluating the alignment of existing maintenance processes with the envisioned 'smart' criteria.

Research question 2: How could the implementation of an agile and smart maintenance process be achieved to ensure transparency, traceability, and enhanced data security compared with other platforms? This research question delves into the practical application of our envisioned smart maintenance framework.

3.3 Research methodology

In light of the challenges underscored by existing literature, this study endeavors to craft a pragmatic framework geared towards the augmentation of maintenance processes within manufacturing companies. The primary focus of this framework lies in the implementation of blockchain technology. Drawing upon the deficiencies spotlighted in the reference text, specifically the absence of systematic communication among stakeholders, inefficient data collection methods, and the pressing need for expediting communication processes, this methodology outlines a meticulous and all-encompassing approach to integrating blockchain for the optimization of maintenance management. Having defined the basics of Industrial 4.0, the blockchain, the maintenance and having reviewed the state of the technology, it's time for the practical stage.

To address research question 1, by scrutinizing the current process, we pinpoint essential attributes, including transparency, traceability, and efficiency. Should the existing practices fall short of these benchmarks, attention pivots towards envisaging a future smart maintenance process. To address these comprehensively, the first step involves modeling the existing process to assess its characteristics. Does this process seamlessly integrate with Industry 4.0 principles, ensuring transparency, and drawing insights from blockchain literature? If not, the next imperative is to

propose a forward-looking maintenance approach that accommodates both the theoretical requisites of a smart process, such as decentralized data sharing, and the practical necessities of industrial maintenance. After conducting a thorough literature review and defining the problem, the existing maintenance process is meticulously delineated and mapped out. Building upon this foundation, a proposed future process is conceptualized and mapped to provide a clear trajectory for advancement.

In response to research question 2, by developing and implementing smart contracts tailored to the characteristics of our newly modeled maintenance process, we aim to automate and enforce key aspects efficiently. The strategic programming of smart contracts will facilitate the seamless execution of tasks, agreements, and interactions relevant to maintenance operations. This comprehensive initiative transcends theoretical boundaries and ventures into the pragmatic realm, ensuring that the proposed smart process is not only conceptualized but also translated into real-world scenarios. Our focus on the comparative analysis between the traditional maintenance process and the blockchain-enhanced process within the framework underscores the practical implications of the proposed smart maintenance approach. This research question serves as a pivotal phase, offering valuable insights and a nuanced understanding of how the suggested agile maintenance process performs in diverse industrial contexts. To accomplish this, it is imperative to establish a programming environment conducive to development, followed by the creation of a smart contract and a user-friendly front-end interface.

CHAPTER 4 PROPOSAL FOR SMART MAINTENANCE

In this chapter, we embark on a comprehensive exploration of the maintenance processes. Leveraging the GRMI4.0 (a guide for representing and modeling Industry 4.0 business processes) framework, we meticulously model the existing maintenance processes. The visual representation provided by GRMI4.0 serves as a powerful tool, revealing communication effectively, and all the steps in the model are clearly visible.

As part of our transformative approach, a new maintenance process is introduced and intricately modeled using GRMI4.0. This proposed process serves as the blueprint for a paradigm shift. GRMI4.0 allows us to vividly capture the enhancements in communication channels, data collection, and traceability that the process is poised to deliver. The visual representation serves as a guide for the subsequent stages of our exploration.

Building upon the insights gained from the proposed process model, we delve into the programming environment and smart contract development. This stage involves translating the conceptualized process into tangible, executable code. The chosen programming environment provides the foundation for the creation of a smart contract, a self-executing digital agreement that automates and enforces the proposed maintenance process. This section elucidates the intricacies of the coding process, emphasizing its alignment with the proposed process model.

To complement the back-end functionality, a front-end interface is crafted, providing a user-friendly interaction point for stakeholders. Leveraging the Application Binary Interface (ABI), it seamlessly connects with the smart contract, ensuring a smooth flow of information between users and the blockchain. A step-by-step demonstration of each stage—from GRMI4.0 modeling to programming, smart contract development, and front-end interface creation—offers a comprehensive view of the entire process evolution, showcasing the synergy between technology and maintenance management.

4.1 Smart maintenance process

4.1.1 Creation path

Define the existing maintenance process: To assess the compatibility of the current maintenance process with technological advancements, this step involves establishing a comprehensive

framework. The aim is to encompass every aspect of the maintenance workflow, starting from its initial stages and extending to its culmination. This process allows for a thorough evaluation of how well the existing maintenance procedures align with the requirements of emerging technologies.

Mapping existing maintenance process: The mapping of the existing maintenance process plays a crucial role in understanding the current workflow and identifying areas of improvement within an organization. This process involves thoroughly examining and documenting the various steps, activities, and stakeholders involved in the maintenance management system. Additionally, mapping the maintenance process helps in visualizing the flow of information, communication channels, and decision-making points, providing a comprehensive overview of the entire maintenance lifecycle. The gathered insights from the mapping exercise serve as a valuable foundation for further research, enabling the formulation of recommendations and the development of an enhanced maintenance process that can contribute to improved resource allocation, cost-effectiveness, and overall operational efficiency.

Define the future process and adding technology: In alignment with research question 1, which aims to enhance the existing process and incorporate technology, this step involves defining the future workflow. With variables identified and insights gained from analyzing the current process, it is now possible to develop an optimized flow chart that seamlessly integrates technological solutions.

Mapping new process: Building upon the analysis of the existing maintenance process and the definition of the future flow, the next crucial step is mapping the new process. This involves designing a comprehensive framework that incorporates the insights and recommendations derived from the previous stages. It also includes considering the integration of technological advancements to enhance efficiency and effectiveness.

4.1.1.1 Define and mapping existing process

GRMI4.0 (a guide for representing and modeling Industry 4.0 business processes) is a new business process representation proposed by Jeremie Mosser that has been used for modeling the processes for maintenance. The selection of the GRMI4.0 model over alternative representations for Industry 4.0 stems from a thorough examination of existing BPM standards. While no single formalism fully met all Industry 4.0 requirements, this evaluation identified GRMI4.0, based on BPMN, as

the most promising. Chosen for its ability to effectively convey the complexities of 4.0 solutions, especially in technology and data representation, GRMI4.0 emerged as the preferred model for our research (Mosser, 2020).

The GRMI 4.0 diagram represents a comprehensive framework for maintenance services, categorized into three main groups: the main company team, the technology aspect, and the third-party providers of maintenance services.

The main company team is represented by various stakeholders, each with their specific roles and responsibilities. The production who start the request for maintain the machines and equipment.. The maintenance department, responsible for planning and executing maintenance activities The approval which basically done by the managers. Lastly, the financial department handles the financial aspects related to maintenance activities.

To enhance the visual clarity and ease of following the flow, each stakeholder and group in the diagram is assigned a unique color. This color scheme helps in distinguishing between different entities and facilitates a better understanding of their involvement in the maintenance process.

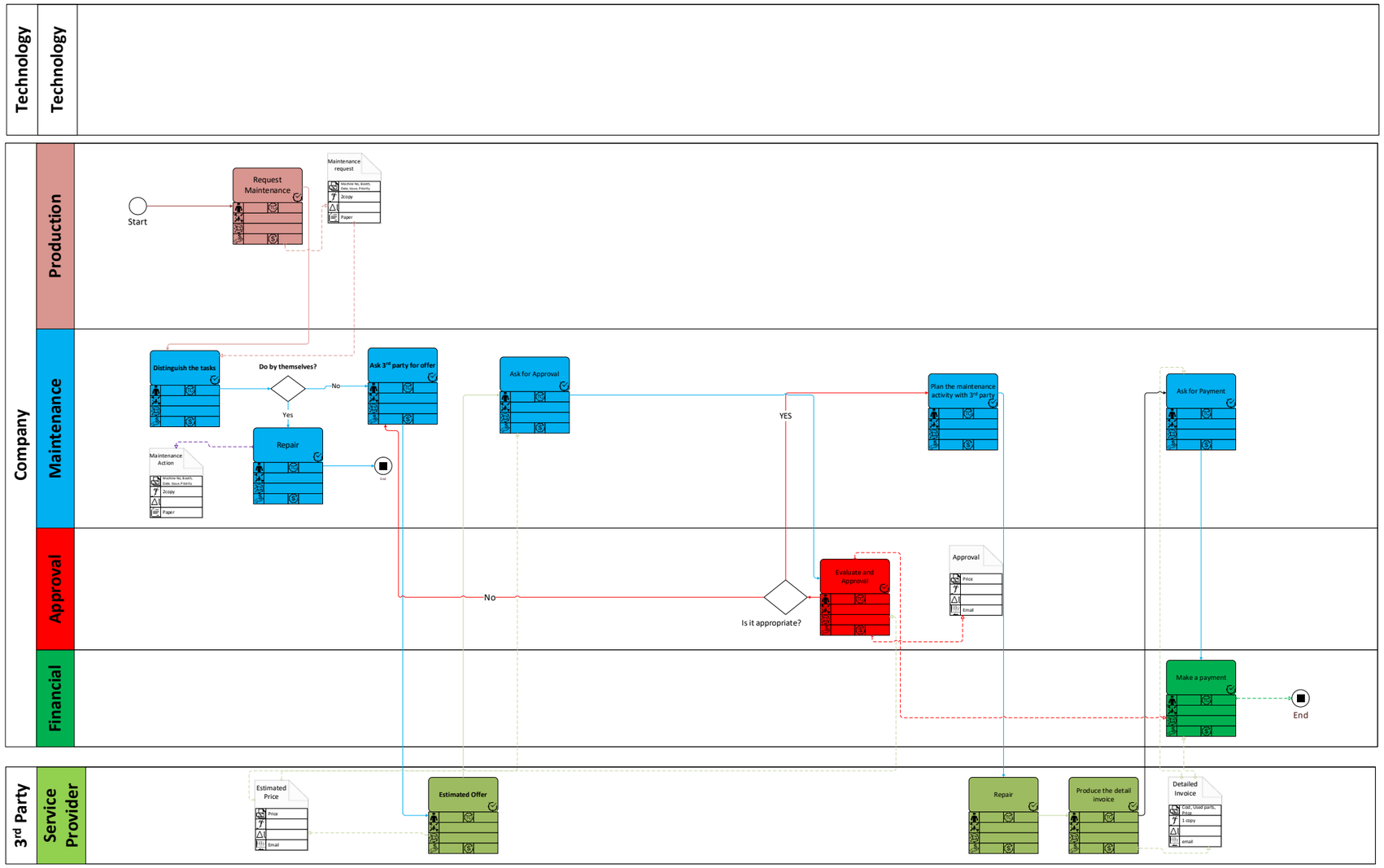
In addition to the main company team, the diagram also encompasses the technology aspect, although its implementation is not currently in use.

Lastly, the third-party maintenance service providers are represented as an essential part of the overall maintenance ecosystem. These external entities may include specialized contractors or vendors who offer specific maintenance services to complement the main company team's efforts. Their inclusion in the diagram emphasizes the collaboration and cooperation required between the main company and external service providers to ensure comprehensive maintenance support. The maintenance process commences with the production staff initiating a maintenance request by completing a designated maintenance work request. Once the request is completed, the maintenance request sends to maintenance team.

The maintenance responsible reviews the request and determines whether the task will be executed by the in-house maintenance team or if it should be outsourced to a third-party service provider. In cases where the maintenance is to be performed by the internal team, the maintenance responsible assigns suitable personnel and schedules the work, accordingly, concluding the process. However, if the maintenance task is delegated to a third-party provider, the maintenance responsible initially requests an investigation into the issue and specifies the preferred timeframe for the service

provider to attend to the company's needs. Simultaneously, the service provider is asked to provide an estimated cost for the maintenance task. The estimated cost is send for approval to managers if the estimated cost is approved, the maintenance responsible proceeds to arrange for the third-party service provider to carry out the maintenance task. In cases where the cost estimate is rejected, the maintenance responsible explores alternative service providers or requests that the initial provider revise their estimated cost accordingly.

Following the completion of the maintenance task by the third-party service provider, an invoice is sent to the maintenance manager via email. Subsequently, the maintenance manager forwards the invoice to the finance department for payment processing, marking the successful conclusion of the maintenance process. For a visual representation of the process flow, refer to Figure 4.1, which depicts the process modeling in GRMI 4.0 for the maintenance process.



Performance		Performance		Remarque(s)	Version	
TC		TC		Notes :	**Organisation**	Welbilt
$		$			**Projet**	Blockchain for Maintenance
					Titre	Existing Maintenance Process
	N.A		N.A		**Auteur**	Salman Momen
	Done By Staff		Done By 3rd Party		**Date**	2023.10.19

Figure 4.1 Existing Process GRMI4.0 Diagram

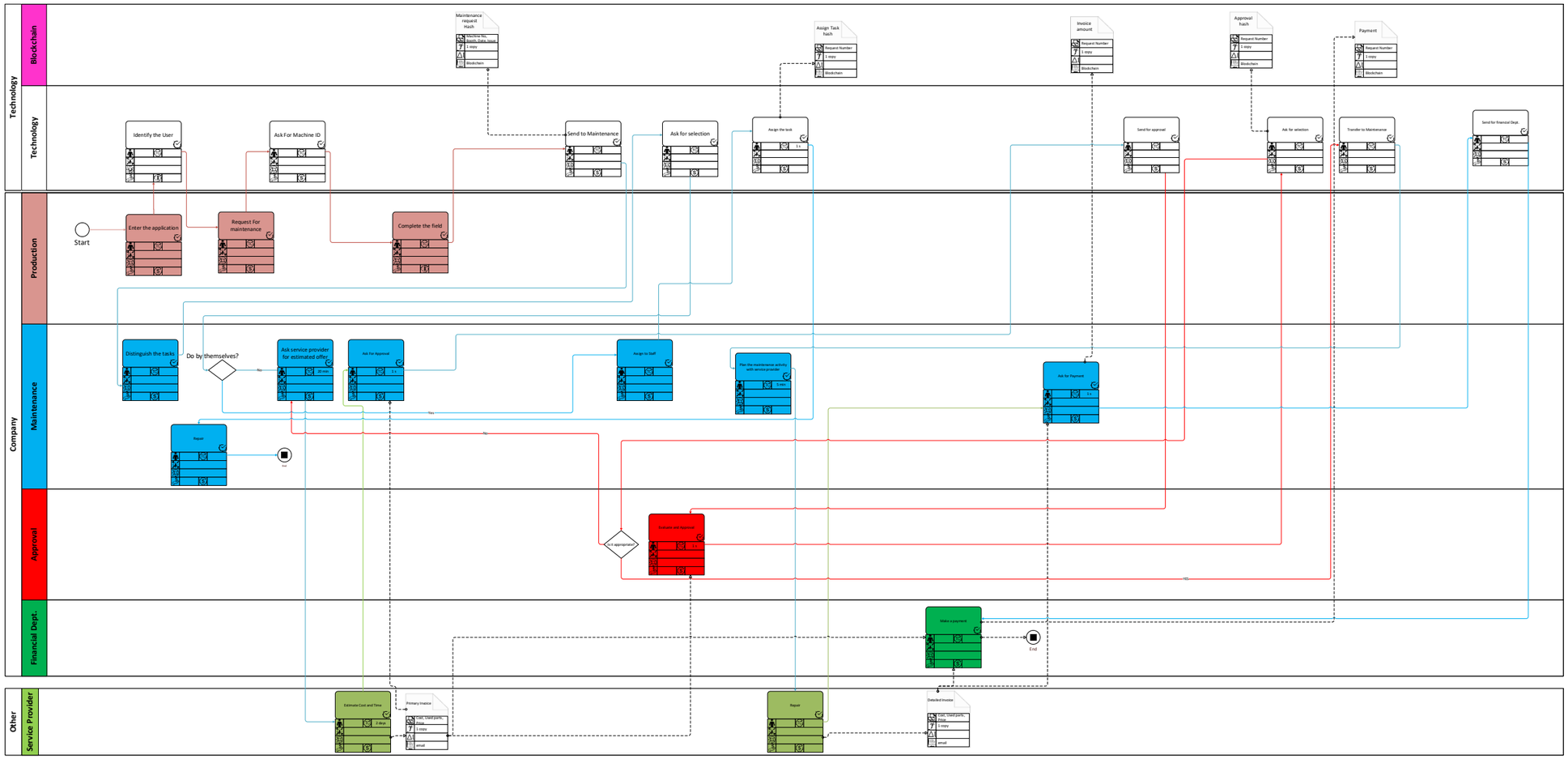
4.1.1.2 Define and mapping future maintenance process

As part of the process enhancement, integrating technology into the maintenance workflow aims to improve efficiency and effectiveness. Building upon insights gained from mapping the existing maintenance process and defining the future flow, mapping the new process involves designing an updated and optimized workflow. This process includes integrating identified technological solutions to optimize various stages of maintenance management.

The mapping exercise visualizes revised steps, activities, and stakeholder roles, considering the utilization of technology at each stage. By mapping the new process, organizations establish a clear blueprint for the implementation of the optimized workflow, ensuring that the integration of technology aligns with the defined future flow. This mapping serves as a guiding framework for transitioning from the existing process to the improved maintenance system, enabling organizations to leverage technology for enhanced operational outcomes. This process is outlined using the GRMI4.0 proposed by Jeremie Mosser.(Mosser, 2020)

Similar to the existing process, the production staff initiates the new process. Each staff has a user and an ID, which may be either his name or a unique number, after filling all necessary information the request sends to maintenance department. We can set where is necessary the details, the date and time are immediately recorded in the blockchain.

The process continues in alignment with the paper-based process, incorporating technology and blockchain. The complete process is illustrated in Figure 4.2 Future Process GRMI4.0 Diagram. Determining which part of the process will be registered on the blockchain and which will not be a crucial consideration.



Performance	Performance	Remarque(s)	Version	
		Notes :	**Organisation**	Weibilt
			Projet	Blockchain for Maintenance
			Titre	Proposed Maintenance Process
			Auteur	Salman Momen
Done By Staff	Done by service provider		**Date**	2023.10.19

Figure 4.2 Future Process GRMI4.0 Diagram

4.1.2 Process comparison

A fundamental distinction between the traditional paper-based maintenance processes and the proposed blockchain-enabled framework lies in operational efficiency and time savings. While paper-based systems often entail manual data entry, retrieval, and verification, the blockchain-enabled framework automates these processes through smart contracts.

The contrast between traditional paper-based processes and the proposed smart maintenance framework lies in their flexibility and adaptability. While paper-based systems are often rigid and static, the blockchain-enabled framework offers unparalleled flexibility to adjust to changing operational needs. This agility fosters a culture of continuous improvement, empowering organizations to refine their maintenance strategies in real-time based on data-driven insights and emerging industry standards.

Another key distinction between the two processes is the level of transparency and traceability offered by technology. While paper-based processes often suffer from opacity and a lack of auditability, technology provides an immutable ledger that records every maintenance transaction in a transparent and tamper-proof manner. This transparency enhances accountability and trust among stakeholders, as each maintenance activity is verifiable and traceable back to its origin.

4.2 Smart maintenance implementation

4.2.1 Implementation steps

Define programming environment: This step includes identifying the most suitable programming environment for the project, one that possesses the necessary functions required for implementing a blockchain and offers sufficient flexibility to accommodate the functionality outlined in the project's flow chart. Selecting an appropriate programming environment is crucial, as it forms the foundation for the entire project.

Create Smart Contract: In line with research question 2, the next step, after mapping the new process, is to develop a smart contract. This crucial phase involves selecting a programming language tailored to the intricacies of the mapped process. The objective is to create an efficient and secure smart contract that reflects the identified process enhancements.

Front End Interface Creation: The front-end application is a critical component of the overall blockchain system, as it provides the interface for users to interact with the system. Through careful design, selection of appropriate technologies, and diligent implementation and testing, the front-end application can deliver a user-friendly and visually appealing experience, enhancing the usability and adoption of the blockchain system.

4.2.2 Smart contract

A smart contract operates as a code-driven consensus, facilitating trustless transactions without intermediaries. Its applications extend beyond cryptocurrencies to sectors such as supply chain, IoT, healthcare, and business process management. With security akin to blockchain data, smart contracts define triggers, participant obligations, rights, and outcomes. The accuracy and fairness of blockchain execution hinge on secure code implementation, emphasizing rigorous testing and professional review for verification (Bhushan et al., 2021).

Upon completion of the new process mapping, the subsequent step entails the creation of a smart contract. The goal is to develop an efficient and secure smart contract that aligns with the identified process enhancements.

The statement "// SPDX-License-Identifier: UNLICENSED" is used as a placeholder in Solidity smart contracts to indicate that the contract does not have a specific open-source license attached to it. It doesn't refer to an actual license type; rather, it signifies that the code is not open-source and is not provided under any recognized license.

Open-source licenses are legal agreements that dictate how the source code can be used, shared, modified, and distributed. By specifying a recognized SPDX license identifier in a Solidity contract, you are clearly indicating the terms under which others can use or modify your code. However, if you use "// SPDX-License-Identifier: UNLICENSED", it indicates that the code is not licensed for public use, modification, or distribution. It essentially means that you are reserving all rights and control over the code and are not granting any permissions for others to use it.

The next step is to define how our programme will be compiled; in this case, we choose to do so using version 8.0 or later; this serves to define a way to compile in case a more recent version

makes changes to some function; this prevents the programme from changing when there is a new update.

OpenZeppelin's "Ownable.sol" and "Pausable.sol" are smart contract libraries that offer essential authorization control functionalities and emergency stop mechanisms for Ethereum-based applications. "Ownable.sol" provides a basic authorization system, allowing a single designated owner to manage access and permissions within a contract, enhancing security and control over the contract's functionality. On the other hand, "Pausable.sol" introduces an emergency stop mechanism, enabling the contract owner to temporarily halt certain operations in case of vulnerabilities, bugs, or unexpected circumstances. By importing and utilizing these libraries, developers can implement secure and controlled smart contracts while maintaining the flexibility to swiftly react to unforeseen events.(Ownable, 2023; Pausable, 2023)

Structures and Variables: The contract declares several variables and a structure `Request`, which is used to keep track of machine maintenance requests. It also creates arrays, mappings, and events for various operational needs. The contract includes a list of staff members and various role-based access control mappings (like supervisor, maintenance manager etc.)

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/security/Pausable.sol";

contract MachineMaintenancetest6 is Ownable, Pausable {
   struct Request {
    string position;
    uint machineNumber;
    bool assigned;
    bool approvedByFactoryManager;
    bool approvedByProductionManager;
    bool completed;
    bool doneByThirdParty;
    bool invoiceIssued;
    uint invoiceAmount;
    bool senttoMaintenanceManager;
    uint internalWorkPercentage;
    uint externalWorkPercentage;
    }
```

```
    // Array to store all requests
    Request[] public allRequests;

    event SupervisorAlerted(address operator);
    event InvoiceIssued(address sender, uint requestIndex, uint invoiceAmount);
    event NewRequestCreated(address indexed requester, uint indexed requestIndex,
string position, uint machineNumber);
    event RequestSentToMaintenance(uint requestIndex);

    mapping (address => Request[]) requests;
    mapping (address => bool) isSupervisor;
    mapping (address => bool) isMaintenanceManager;
    mapping (address => bool) isFactoryManager;
    mapping (address => bool) isProductionManager;
    mapping (address => bool) isThirdParty;
    address[] public staff;

    address public maintenanceManager;
    address public factoryManager;
    address public supervisor;
    address public thirdParty;
    address public productionManager;

    uint internalWorkPercentage;
    uint externalWorkPercentage;
    uint internalPercentage = getInternalWorkPercentage();
    uint externalPercentage = getExternalWorkPercentage();
```

Modifiers such as onlySupervisor() and onlyMaintenanceManager() play a pivotal role in smart contracts by confining the execution of functions according to specific user roles, such as supervisors or maintenance managers. They act as conditional gatekeepers, altering the way functions within the contract operate, ensuring that only authorized individuals can trigger them.

```
  modifier onlySupervisor() {
        require(msg.sender == supervisor, "Only supervisor can call this
function.");
        _;
    }

    modifier onlyMaintenanceManager() {
        require(msg.sender == maintenanceManager, "Only maintenance manager can
call this function.");
        _;
```

```
    }

    modifier onlyFactoryManager() {
        require(msg.sender == factoryManager, "Only factory manager can call this
function.");
        _;
    }

    modifier onlyProductionManager() {
        require(msg.sender == productionManager, "Only production manager can
call this function.");
        _;
    }

    modifier onlyThirdParty() {
        require(msg.sender == thirdParty, "Only third party can call this
function.");
        _;
    }

    modifier onlyStaff() {
    require(isStaff(msg.sender), "Only staff members can call this function.");
    _;
    }

    modifier onlyMaintenanceManagerOrFactoryManager() {
    require(msg.sender == maintenanceManager || msg.sender == factoryManager,
"Only maintenance manager or factory manager can call this function.");
    _;
    }
```

The constructor function, a distinctive element, is invoked upon the deployment of the contract, operating as an initiation point. Within this context, its role involves establishing designated addresses for crucial entities like the Maintenance Manager, Factory Manager, Supervisor, Third Party, and Production Manager.

```
 constructor(address _maintenanceManager, address _factoryManager, address
_supervisor, address _thirdParty, address _productionManager) {
    maintenanceManager = _maintenanceManager;
    factoryManager = _factoryManager;
    supervisor = _supervisor;
    thirdParty = _thirdParty;
    productionManager = _productionManager;
    }
```

Within the scope of this contract, the functions `getInternalWorkPercentage()` and `getExternalWorkPercentage()` hold significance as view functions. These functions facilitate the retrieval of their corresponding percentages without inducing any alterations to the contract's state. Consequently, they are categorized and designated as `view` functions, ensuring their read-only nature.

```
  function getInternalWorkPercentage() public view returns (uint) {
    return internalWorkPercentage;
    }

    function getExternalWorkPercentage() public view returns (uint) {
    return externalWorkPercentage;
    }
```

In the context of this contract, a group of imperative set functions including `setMaintenanceManager()`, `setFactoryManager()`, `setSupervisor()`, `setThirdParty()`, and `setProductionManager()`, serve the pivotal purpose of modifying addresses associated with distinct roles. An inherent stipulation, enforced through the `onlyOwner` modifier, dictates that these functions can solely be invoked by the contract's owner, ensuring a controlled and secure alteration of roles.

```
function setMaintenanceManager(address _maintenanceManager) public onlyOwner {
    maintenanceManager = _maintenanceManager;
    }

    function setFactoryManager(address _factoryManager) public onlyOwner {
    factoryManager = _factoryManager;
    }

    function setSupervisor(address _supervisor) public onlyOwner {
    supervisor = _supervisor;
    }

    function setThirdParty(address _thirdParty) public onlyOwner {
    thirdParty = _thirdParty;
    }

    function setProductionManager(address _productionManager) public onlyOwner {
    productionManager = _productionManager;
    }
```

The "Request Maintenance Function," denoted by the `requestMaintenance()` function, plays a pivotal role in the smart contract's operational framework. This function empowers authorized staff members to generate fresh maintenance requests, thereby initiating a streamlined maintenance workflow. The function is meticulously designed to ensure that only staff members possess the privilege to trigger it. Upon invocation, the function instantiates a new maintenance request, encompassing crucial details such as the position and machine number. These requests subsequently traverse a comprehensive progression, encompassing stages like assignment, approval, completion, and invoicing, each meticulously tracked by the contract's inherent attributes.

```
 function requestMaintenance(string memory position, uint machineNumber) public
onlyStaff returns (uint) {
    require(isStaff(msg.sender), "Only staff members can request maintenance.");

    Request memory newRequest = Request({
        position: position,
        machineNumber: machineNumber,
        senttoMaintenanceManager: false,
        assigned: false,
        approvedByFactoryManager: false,
        approvedByProductionManager: false,
        completed: false,
        doneByThirdParty: false,
        invoiceIssued: false,
        invoiceAmount: 0,
        internalWorkPercentage: 0,
        externalWorkPercentage: 0
    });

     allRequests.push(newRequest);
    uint newIndex = allRequests.length - 1; // Get the index of the newly created
request
    emit NewRequestCreated(msg.sender, newIndex, position, machineNumber); //
Emit the event with the request number

    return newIndex; // Return the index of the newly created request
    }
```

The function `sendToMaintenanceManager(uint requestIndex)` plays a pivotal role within the smart contract's operational framework, enabling supervisors to transmit maintenance requests to the designated maintenance manager. Accessible solely to supervisors, this function triggers a modification in a critical state variable within the Request struct, indicating the dispatch of the request to the maintenance manager. The act of invoking this function effectively shifts the status of a particular request, denoting its transmission to the maintenance manager.

```
   function sendToMaintenanceManager(uint requestIndex) public onlySupervisor {
        require(requestIndex < allRequests.length, "Invalid request index");

        // Mark the request as sent to maintenance manager
        allRequests[requestIndex].senttoMaintenanceManager = true;

        emit RequestSentToMaintenance(requestIndex);
    }
```

The "Get Request Data Function," which is represented in the getRequestData() function, is a critical component of the smart contract's architecture. Based on its index, this function serves as a gateway to vital request data. It permits the retrieval of intricate facts wrapped within a specific request when invoked, contributing to a transparent and informed overview of the maintenance workflow. This function employs a robust validation mechanism to protect against erroneous data access through cautious implementation.

```
 //Function to retrieve request data by its number
   function getRequestData(uint requestIndex) public view returns (string memory,
uint, bool, bool, bool, bool, bool, bool, uint) {
    require(requestIndex < allRequests.length, "Invalid request index");

    Request storage currentRequest = allRequests[requestIndex];
    return (
        currentRequest.position,
        currentRequest.machineNumber,
        currentRequest.assigned,
        currentRequest.approvedByFactoryManager,
        currentRequest.approvedByProductionManager,
        currentRequest.completed,
        currentRequest.doneByThirdParty,
        currentRequest.invoiceIssued,
        currentRequest.invoiceAmount
    );
```

```
    }
```

In the context of this smart contract, the dynamic coordination of task distribution and assignment is facilitated through two fundamental functions: `setWorkDistribution()` and `decideWorkAssignment()`. These functions collaboratively underpin the systematic allocation of responsibilities and the determination of the designated party to undertake the given task.

The `setWorkDistribution()` function serves as a cornerstone, enabling the establishment of a well-calibrated division of labor between internal and external entities. When invoked, this function empowers the maintenance manager to precisely define the proportion of work that each party—internal and external—will assume for a specific request. Notably, robust validation mechanisms ensure the accuracy of the distribution percentages, with their sum mandated to be exactly 100. Through this precise mechanism, the contract fosters transparency and accountability in the work distribution process.

Meanwhile, the `decideWorkAssignment()` function takes center stage as the decision-making authority for task allocation. Exclusively under the purview of the maintenance manager, this function clarifies task allocation by determining the responsible party based on the predefined distribution percentages. By astutely evaluating the external work percentage, this function determines whether the task is delegated to a third party or retained within the internal domain.

```
  function setWorkDistribution(uint requestIndex, uint internalPercentage, uint
externalPercentage) public onlyMaintenanceManager {
    require(requestIndex < allRequests.length, "Invalid request index");
    require(internalPercentage + externalPercentage == 100, "Work distribution
should sum up to 100.");

    allRequests[requestIndex].internalWorkPercentage = internalPercentage;
    allRequests[requestIndex].externalWorkPercentage = externalPercentage;
    }


    function decideWorkAssignment(uint requestIndex) public
onlyMaintenanceManager {
    Request storage currentRequest = allRequests[requestIndex];
    require(!currentRequest.assigned, "Task already assigned.");

    if (currentRequest.externalWorkPercentage > 0) {
```

```
        currentRequest.assigned = true;
        currentRequest.doneByThirdParty = true;
    } else {
        currentRequest.assigned = true;
        currentRequest.doneByThirdParty = false;
    }
    }
```

The harmonious progression of task completion and the intricate invoicing process is skillfully managed through a quartet of essential functions within this smart contract: `completeTask()`, `sendInvoice()`, `approveInvoice()`, and `payInvoice()`. These functions collectively underpin the seamless finalization of maintenance tasks and the meticulous invoicing cycle, encapsulating pivotal stages in the contract's operational flow.

The `completeTask()` function, carefully executed under the aegis of either the maintenance manager or the factory manager, marks a pivotal juncture. This function, upon activation, validates the task's assignment status and ensures that the task has not already been completed. By setting the completion status to "true," the function affirms the successful culmination of the maintenance activity, engendering clarity in the contract's workflow.

Simultaneously, the `sendInvoice()` function, exclusive to the third party, exercises authoritative control over the invoicing process. Through this function, third parties are empowered to issue invoices for tasks they have completed. Rigorous validation ensures that the task has indeed been accomplished by a third party and that no prior invoice has been generated. Upon satisfying these conditions, the function records the invoiced amount and raises an "InvoiceIssued" event, thereby attesting to the transparent record-keeping embedded within the contract.

Upon the inception of an invoice, the `approveInvoice()` function, exclusive to factory managers, guides a critical assessment. By meticulously evaluating the invoice amount, this function offers an either/or scenario for approval: if the invoice amount exceeds 10,000 units, factory managers wield the authority to approve the invoice. Otherwise, this authority is vested in production managers. This bifurcated approach ensures an efficient, balanced review process, further validating the contract's rigorous adherence to accountability and transparency.

Completing the cycle, the `payInvoice()` function, exclusive to the contract owner, pertains to the monetary aspect of the invoicing process. Once prerequisites are met, including the invoice's

approval and the fulfillment of the invoiced amount by the sender, this function facilitates the seamless transfer of payment to the third party.

```
 function completeTask(uint requestIndex) public
onlyMaintenanceManagerOrFactoryManager {
    Request storage currentRequest = allRequests[requestIndex];
    require(currentRequest.assigned, "Task not yet assigned.");
    require(!currentRequest.completed, "Task already completed.");
    currentRequest.completed = true;
    }


    function sendInvoice(uint requestIndex, uint amount) public onlyThirdParty {
    require(requestIndex < allRequests.length, "Invalid request index");

    Request storage currentRequest = allRequests[requestIndex];
    require(currentRequest.doneByThirdParty, "Not a third party task.");
    require(!currentRequest.invoiceIssued, "Invoice already issued.");

    currentRequest.invoiceAmount = amount;
    currentRequest.invoiceIssued = true;
    emit InvoiceIssued(msg.sender, requestIndex, amount);
}


    function approveInvoice(uint requestIndex) public onlyFactoryManager {
    require(requestIndex < allRequests.length, "Invalid request index");

    Request storage currentRequest = allRequests[requestIndex];
    require(currentRequest.doneByThirdParty, "Not a third party task.");
    require(currentRequest.invoiceIssued, "Invoice not issued yet.");
    require(!currentRequest.approvedByFactoryManager &&
!currentRequest.approvedByProductionManager, "Invoice already approved.");

    if (currentRequest.invoiceAmount > 10000) {
        currentRequest.approvedByFactoryManager = true;
    } else {
        currentRequest.approvedByProductionManager = true;
    }
}


    function payInvoice(uint requestIndex) public payable onlyOwner {
        Request storage currentRequest = allRequests[requestIndex];
        require(currentRequest.doneByThirdParty, "Not a third party task.");
```

```
        require(currentRequest.invoiceIssued, "Invoice not issued yet.");
        require(currentRequest.approvedByFactoryManager ||
currentRequest.approvedByProductionManager, "Invoice not approved.");
        require(msg.value >= currentRequest.invoiceAmount, "Insufficient payment
amount.");

        // Transfer the payment to the third party
        payable(thirdParty).transfer(currentRequest.invoiceAmount);
    }
```

Facilitating the establishment of role-based designations, the quartet of functions - `addSupervisor()`, `addMaintenanceManager()`, `addProductionManager()`, and `addThirdParty()` - serves as a pivotal feature within this smart contract. Empowering the factory manager with the authority to meticulously assign addresses to specific roles, these functions introduce a layer of granularity to the contract's operational structure.

The `addSupervisor()` function, accessible solely to the factory manager, seamlessly embeds the assigned address into the role of a supervisor. By invoking this function, the factory manager effectively designates an individual as a supervisor, thus delineating their responsibility within the system. Correspondingly, the `addMaintenanceManager()` function brings forth the analogous functionality, enabling the factory manager to designate an address as the maintenance manager. This intricate web of function-crafted roles intricately interlocks, reinforcing the contract's robustness.

Moreover, the `addProductionManager()` function enables the factory manager to designate an address as the production manager, aligning this essential role within the contract's operational hierarchy. Lastly, the `addThirdParty()` function empowers the factory manager to nominate an address as a third party, further enriching the roster of designated entities within the contract. Through this systematic ensemble of functions, the smart contract not only endows the factory manager with role-allocation authority but also solidifies the contract's adaptability and flexibility in accommodating distinct roles and their associated addresses.

```
    function addSupervisor(address _supervisor) public onlyFactoryManager {
        supervisor = _supervisor;
        isSupervisor[_supervisor] = true;
```

```
    }

    function addMaintenanceManager(address _maintenanceManager) public
onlyFactoryManager {
        maintenanceManager = _maintenanceManager;
        isMaintenanceManager[_maintenanceManager] = true;
    }

    function addProductionManager(address _productionManager) public
onlyFactoryManager {
        productionManager = _productionManager;
        isProductionManager[_productionManager] = true;
    }

    function addThirdParty(address _thirdParty) public onlyFactoryManager {
        thirdParty = _thirdParty;
        isThirdParty[_thirdParty] = true;
    }
```

The "Supervisor Alert Function," encapsulated within the `alertSupervisor()` function, operates as an intrinsic mechanism designed to emit an event that serves as a notification to the supervisor. Operating at an internal level, this function enforces the stipulation that the supervisor's role must be designated. While the core logic for alerting the supervisor could potentially be implemented externally, the smart contract lays the foundation for such notifications through the emission of a dedicated event, exemplifying the contract's potential for collaborative, off-chain integrations.

```
   function alertSupervisor(address operator) internal {
        require(isSupervisor[supervisor], "Supervisor not set.");
        // Additional logic to alert the supervisor could be done outside the
contract (just a msg.json file as use case)

        emit SupervisorAlerted(operator);
    }
```

Within the framework of this smart contract, a suite of essential functions—namely, `isStaff()`, `addStaff()`, and `removeStaff()`—comprises a cornerstone for staff management, exemplifying the contract's dynamic control over staff member lists and the subsequent maintenance of these lists.

The `addStaff()` function, an integral facet, places the factory manager in the driver's seat by enabling the addition of staff members to the existing roster. Exclusively accessible to the factory manager, this function orchestrates the inclusion of staff members' addresses into the designated array, enriching the composition of the workforce.

Further reinforcing the contract's personnel management capability, the `isStaff()` function empowers any entity to verify staff membership status. Operating as a view function, it iterates through the staff list, validating whether a given address is indeed a part of the staff. By offering a clear-cut verification mechanism, this function underpins the transparency and reliability of the staff roster.

The `removeStaff()` function, the third vital component, grants the factory manager the authority to execute staff list management at a granular level. This function equips the factory manager with the means to expunge specific staff members from the array, ensuring precision and customization. Through a meticulously orchestrated process, it shifts array elements to cover the void left by the removed staff member and subsequently trims the array length. In sum, these functions collectively bolster the smart contract's operational dynamics, introducing a systematic methodology for staff management, verification, and modification, and thus enhancing the contract's overall efficacy.

```
function addStaff(address wallet) public onlyFactoryManager {
staff.push(wallet);
}

function isStaff(address wallet) public view returns (bool) {
for (uint i = 0; i < staff.length; i++) {
    if (staff[i] == wallet) {
        return true;
    }
}
return false;
}

function removeStaff(address wallet) public onlyFactoryManager {
for (uint i = 0; i < staff.length; i++) {
    if (staff[i] == wallet) {
        // Remove the staff member from the array by shifting elements
        for (uint j = i; j < staff.length - 1; j++) {
            staff[j] = staff[j + 1];
```

```
            }
            // Reduce the length of the array by 1
            staff.pop();
            return;
        }
    }
    }
```

In conclusion, the development and elucidation of various components within the smart contract form a foundational framework that complies with new maintenance operations. The orchestrated interplay of roles, functions, and state variables establishes a dynamic ecosystem wherein distinct user roles are endowed with well-defined privileges and responsibilities. Through the judicious employment of modifiers such as `onlySupervisor()` and `onlyFactoryManager()`, the contract safeguards the integrity of its operations by confining specific functions to authorized personnel. This role-based structure bolsters accountability and transparency, ensuring that tasks are performed within the confines of designated roles.

Furthermore, the smart contract's meticulous structuring of data storage through structures, arrays, and state variables provides a robust foundation for tracking the lifecycles of maintenance requests. The seamless interaction between functions, such as `requestMaintenance()`, `completeTask()`, and `sendInvoice()`, intricately choreographs the progression of requests from initiation to completion. The intricate interplay between state variables and functions reinforces the contract's ability to maintain a coherent and transparent record of tasks, approvals, and invoicing processes.

4.2.3 Front end interface

The interface serves as a bridge between external applications or services and the smart contract, enabling seamless communication and interaction. It provides a set of functions and data structures that define the interface's capabilities and how external entities can interact with the underlying smart contract.

By utilizing the ABI (Application Binary Interface), the interface abstracts the complexity of the smart contract's underlying code and exposes a simplified and standardized interface for external entities to interact with. The ABI specifies the function signatures, input parameters, and return types, ensuring that external applications can correctly communicate with the smart contract.

This interface enables developers to build applications that leverage the functionalities of the smart contract without requiring in-depth knowledge of its internal implementation. It promotes interoperability and extensibility by allowing different systems to communicate and exchange data with the smart contract in a standardized manner.

Through the interface, external applications can invoke functions, retrieve data, and receive event notifications from the smart contract. This facilitates the integration of the smart contract into various decentralized applications (dApps), web services, or any other systems that need to interact with the contract. The interface acts as a contract "blueprint" that outlines the available operations and their usage, ensuring seamless and secure interaction with the underlying smart contract functionality.

The designed application consists of nine tabs, each serving a specific purpose in the context of machine maintenance management. These tabs provide a user-friendly interface for interacting with the underlying smart contract and facilitate the seamless execution of various maintenance-related tasks. The main page shown in Figure 4.3. Let's explore each tab in detail:

Figure 4.3 Front End Interface

1. The "Request Maintenance" tab allows users, who are staff members, to submit maintenance requests for machines. Users can input details such as the position of the machine and machine number of the maintenance request. This tab ensures an organized and efficient process for users to report maintenance issues. We can see the Request Maintenance tab in Figure 4.4.

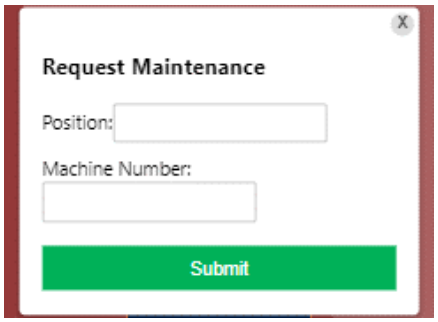


Figure 4.4 Request Maintenance

2. The "Send to Maintenance Manager" tab is dedicated to the supervisor, who holds the responsibility of overseeing maintenance operations. Once a maintenance task is completed, the supervisor can use this tab to send the task details to the maintenance manager for further action. This tab helps streamline the workflow by enabling the supervisor to delegate tasks to the relevant personnel promptly. The Send to Maintenance Manager tab shown in Figure 4.5.

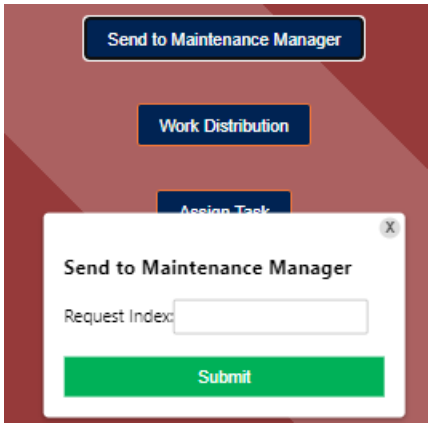


Figure 4.5 Send to Maintenance Manager

3. The "Work Distribution" tab is exclusive to the maintenance manager, who can decide which task could be done by themselves or he can ask from the third party to do this task. This tab includes the percentage which could assign it internally or external. You can see this tab in Figure 4.6.

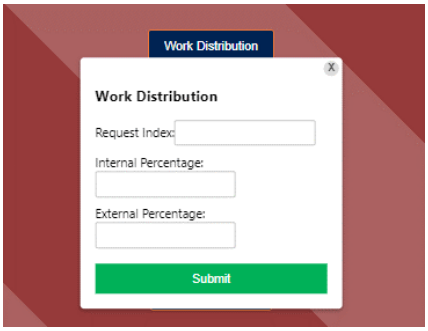


Figure 4.6 Work Distribution

4. The "Assign Tasks" tab is exclusive to the maintenance manager, who can assign tasks to staff members. By utilizing this tab, the maintenance manager can efficiently distribute the workload and ensure that tasks are appropriately assigned based on expertise and availability. This tab enhances coordination and efficiency within the maintenance team. The Assign Task tab shown in Figure 4.7.

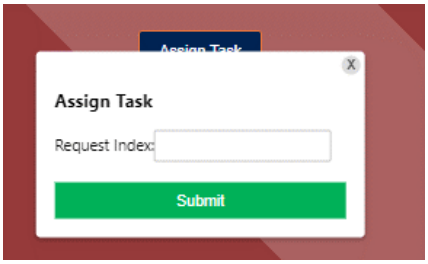


Figure 4.7 Assign Task

5. The "Send Invoice" tab is specifically designed for third-party service providers. Upon completing a maintenance task, they can generate and send an invoice for the services rendered. This tab simplifies the invoicing process and enables seamless communication between the third-party providers and the maintenance management system. You can see this tab in Figure 4.8.

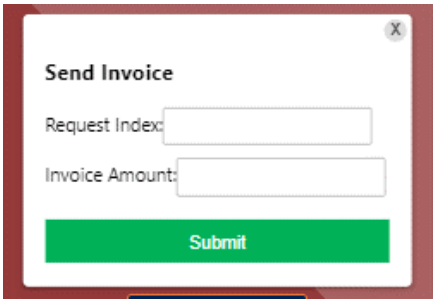


Figure 4.8 Send Invoice

6. The "Approve Invoice" tab is accessible to the factory manager, who holds the authority to review and approve invoices. Based on specific criteria, such as invoice amount, the factory manager can approve or reject the invoices submitted by third-party providers. This tab introduces a crucial layer of financial control and accountability within the maintenance management system. Figure 4.9 illustrated this tab.

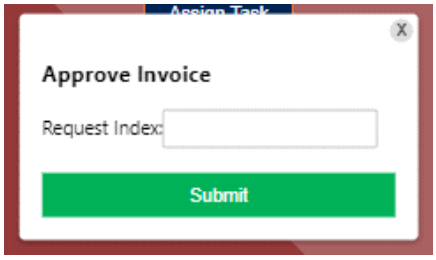


Figure 4.9 Approve Invoice

7. The "Pay Invoice" tab facilitates the payment process for approved invoices. Users, typically the factory manager or authorized personnel, can initiate the payment by specifying the request index and the payment amount. This tab ensures timely and accurate payment to third-party service providers, promoting a smooth financial transaction flow. You can see the Pay Invoice tab in Figure 4.10.

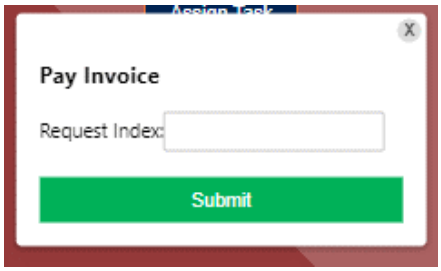


Figure 4.10 Pay Invoice

8. The "Complete Task" tab allows the maintenance manager or factory manager to mark a task as completed once it has been fulfilled. By indicating the completion date and time, this tab serves as a reliable record-keeping mechanism and enables effective tracking of maintenance tasks' status. Figure 4.11 shown the Complete Task tab.

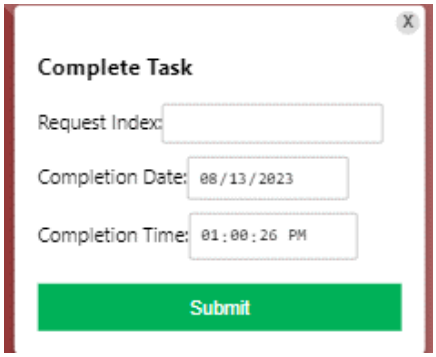


Figure 4.11 Complete Task

9. In the check request maintenance data you can track and trace the maintenance request status by importing the index for the maintenance request. As an example, for this item, we insert the index 1 (Figure 4.12) and after you click on the fetch data you can see all fields of that request (Figure 4.13)

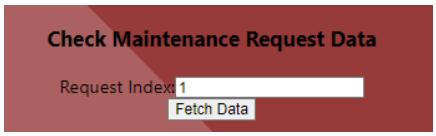


Figure 4.12 Check maintenance data

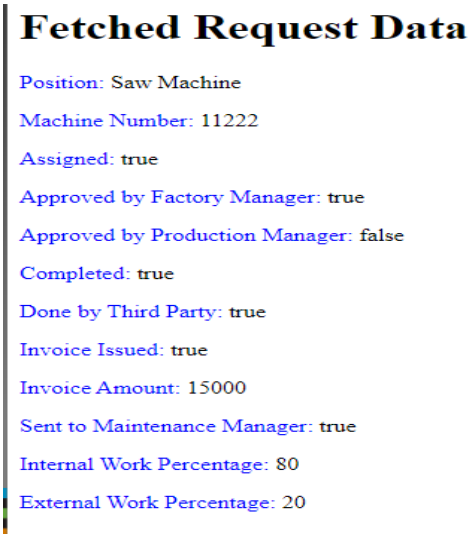


Figure 4.13 Request Status

Overall, the application's design, with these nine tabs provides a comprehensive solution for connecting the smart contract with a user-friendly application. It enhances communication, streamlines workflows, and promotes transparency and accountability throughout the maintenance process. Users can navigate through the tabs seamlessly, performing their respective tasks with ease and efficiency.

CHAPTER 5 VALIDATION

In this pivotal chapter, we delve into the intricate details of the maintenance processes and present a comparison between the existing and proposed methodologies. Beginning with an in-depth exploration of the current maintenance process, we scrutinize its intricacies, strengths, and limitations. The subsequent section unfolds the proposed process, meticulously designed to address the identified gaps and leverage the transformative potential of technology. As we navigate through the implementation phase, our focus sharpens on the integration of smart contracts and the development of an intuitive interface within the framework of the proposed process. Finally, we unveil the results, providing an analysis of the outcomes derived from the application of our novel smart maintenance approach.

5.1 Processes and case study

In conducting a comprehensive evaluation, we initiated a test to assess the efficiency of the existing paper-based maintenance process within the case study company. Staff members engaged in the maintenance activities were involved in the test, which primarily focused on capturing the time required for each step in the traditional workflow. This manual recording process aimed to provide insights into the temporal aspects of the maintenance tasks and identify potential bottlenecks or delays. The gathered data serves as a baseline for the subsequent comparison against the envisioned smart maintenance process, emphasizing the need for streamlined operations and enhanced time management.

5.1.1 Existing process

After several meetings with the management teams in the WELBILT company, the existing process has been defined. Our focus is on the breakdown maintenance process in the company.

The GRMI 4.0 diagram represents a comprehensive framework for maintenance services, categorized into three main groups: the main company team, the technology aspect, and the third-party providers of maintenance services.

The main company team is represented by various stakeholders, each with their specific roles and responsibilities. The factory manager oversees the overall operations, ensuring smooth functioning

of the manufacturing processes. The production manager focuses on optimizing production output and quality. The maintenance manager leads the maintenance department, responsible for planning and executing maintenance activities. The maintenance team consists of skilled technicians and engineers who carry out maintenance tasks on the equipment. The operator operates the machinery on a day-to-day basis, while the supervisor provides guidance and ensures compliance with safety and operational protocols. Lastly, the financial department handles the financial aspects related to maintenance activities.

To enhance the visual clarity and ease of following the flow, each stakeholder and group in the diagram is assigned a unique color. This color scheme helps in distinguishing between different entities and facilitates a better understanding of their involvement in the maintenance process.

In addition to the main company team, the diagram also encompasses the technology aspect, although its implementation is not currently in use.

Lastly, the third-party maintenance service providers are represented as an essential part of the overall maintenance ecosystem. These external entities may include specialized contractors or vendors who offer specific maintenance services to complement the main company team's efforts. Their inclusion in the diagram emphasizes the collaboration and cooperation required between the main company and external service providers to ensure comprehensive maintenance support. You can seen the categorizing in Figure 5.1.

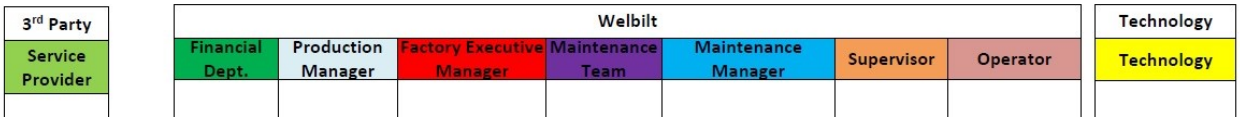


Figure 5.1 Categorizing.

The maintenance process commences with the operator initiating a maintenance request by completing a designated maintenance work request form Figure 5.2.

Figure 5.2 Work order Paper.

This form includes essential details such as the initiator's name, the specific machine or equipment requiring maintenance, the date and time of the request for future traceability, the department and booth where the equipment is located, a concise summary of the identified issue prepared by the operator, and the priority level assigned to the task. The priority levels are divided into three categories: Priority 1 indicates situations involving safety risks, machine breakdowns, or production standstills; Priority 2 suggests that maintenance can be conducted within the next 8 hours; and Priority 3 signifies non-emergency issues, allowing the maintenance team to schedule the work for the next production stoppage.

Once completed, the maintenance request paper is delivered to the production supervisor, who subsequently submits it to the maintenance office.

The maintenance manager reviews the request and determines whether the task will be executed by the in-house maintenance team or if it should be outsourced to a third-party service provider. In cases where the maintenance is to be performed by the internal team, the manager assigns suitable personnel and schedules the work, accordingly, concluding the process. However, if the maintenance task is delegated to a third-party provider, the maintenance manager initially requests an investigation into the issue and specifies the preferred timeframe for the service provider to

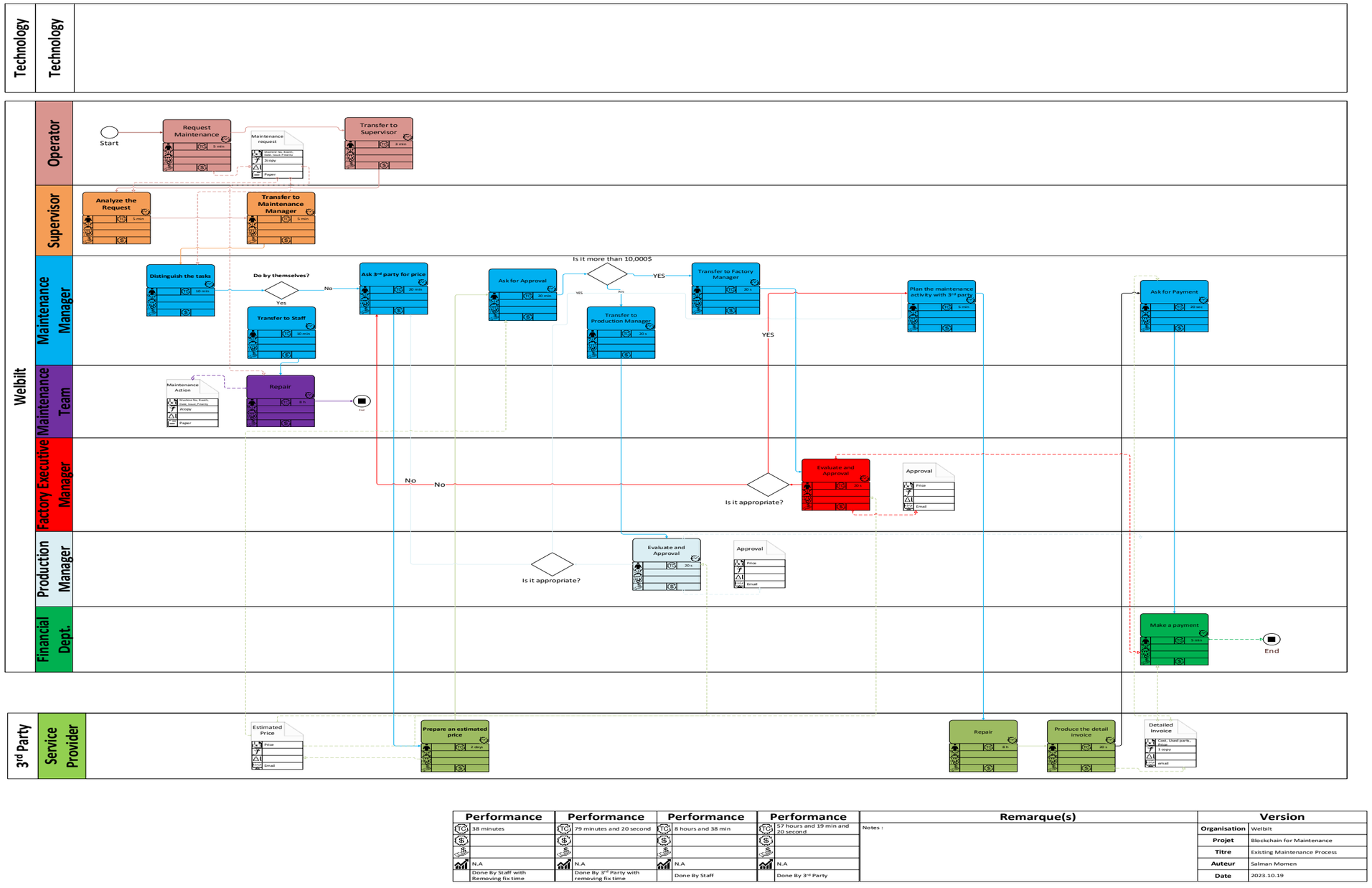
attend to the company's needs. Simultaneously, the service provider is asked to provide an estimated cost for the maintenance task. The estimated cost is analyzed by either the company manager for amounts exceeding $10,000 or the production manager for amounts below that threshold. If the estimated cost is approved, the maintenance manager proceeds to arrange for the third-party service provider to carry out the maintenance task. In cases where the cost estimate is rejected, the manager explores alternative service providers or requests that the initial provider revise their estimated cost accordingly.

Following the completion of the maintenance task by the third-party service provider, an invoice is sent to the maintenance manager via email. Subsequently, the maintenance manager forwards the invoice to the finance department for payment processing, marking the successful conclusion of the maintenance process. For a visual representation of the process flow, refer to Figure 5.3, which depicts the process modeling in GRMI 4.0 for the maintenance process. (Mosser, 2020)

As previously stated, relying on a paper-based process can lead to inefficiencies and complications. The time it takes to complete the necessary paperwork and locate a supervisor can cause delays and disrupt production. Furthermore, supervisors may need to physically go to the maintenance office or make a call to the maintenance manager, adding further delay. Keeping track of maintenance history and tracing the process can also be a challenging and time-consuming task. To ensure smooth production and minimize machine downtime, it is essential to have optimized maintenance processes in place. This is particularly true in the era of Industry 4.0, where all manufacturing facilities are closely interrelated and highly dependent on one another. Particularly in modern environments, multiple parties and departments may be involved in a production scenario and may be interested in an optimal trade-off between having a healthy machine and minimizing maintenance costs (Stodt et al., 2019).

In addition to the aforementioned considerations, an essential aspect to address in this context is the security and management of data. To mitigate these concerns effectively, a blockchain-based platform is proposed as a solution for the maintenance operations within the company. By leveraging the inherent features of blockchain technology, such as decentralization, immutability, and cryptographic security, the platform can ensure the integrity and confidentiality of maintenance-related data. Blockchain's distributed ledger system enables transparent and tamper-proof recording of maintenance activities, enhancing accountability and traceability. Furthermore,

the utilization of smart contracts on the blockchain platform can automate and enforce predefined rules and agreements, streamlining the maintenance process and minimizing the risk of human error or unauthorized access. Implementing a blockchain-based platform for maintenance operations not only addresses data security concerns but also promotes trust, efficiency, and reliability throughout the company's maintenance ecosystem.



Performance	Performance	Performance	Performance	Remarque(s)	Version	
TC: 38 minutes	TC: 79 minutes and 20 second	TC: 8 hours and 38 min	TC: 57 hours and 19 min and 20 second	Notes :	**Organisation**	Welbilt
N.A	N.A	N.A	N.A		**Projet**	Blockchain for Maintenance
Done By Staff with Removing fix time	Done By 3rd Party with removing fix time	Done By Staff	Done By 3rd Party		**Titre**	Existing Maintenance Process
					Auteur	Salman Momen
					Date	2023.10.19

Figure 5.3 Existing Process GRMI4.0 Diagram

5.1.2 Proposed process

Subsequently, leveraging the insights gained from the paper-based maintenance test, we meticulously modeled and implemented the proposed smart maintenance process, infused with cutting-edge technology and blockchain principles. The objective was not only to improve efficiency but also gauge the time required for each stage of the updated workflow. Through systematic data collection and analysis, we aim to provide a comparative evaluation of traditional versus technology-enhanced maintenance processes. This comparative analysis will offer a nuanced understanding of the temporal benefits and operational enhancements achieved through the integration of blockchain technology and smart contracts in the maintenance domain.

The maintenance process is a vital component of industrial operations, ensuring the longevity and efficiency of machinery and systems. In our study, we meticulously examined the existing maintenance protocols within our company, focusing on processes conducted both internally by company staff and externally by third-party entities. Recognizing the importance of time efficiency, our analysis delved into detailed time measurements for each step of the maintenance process.

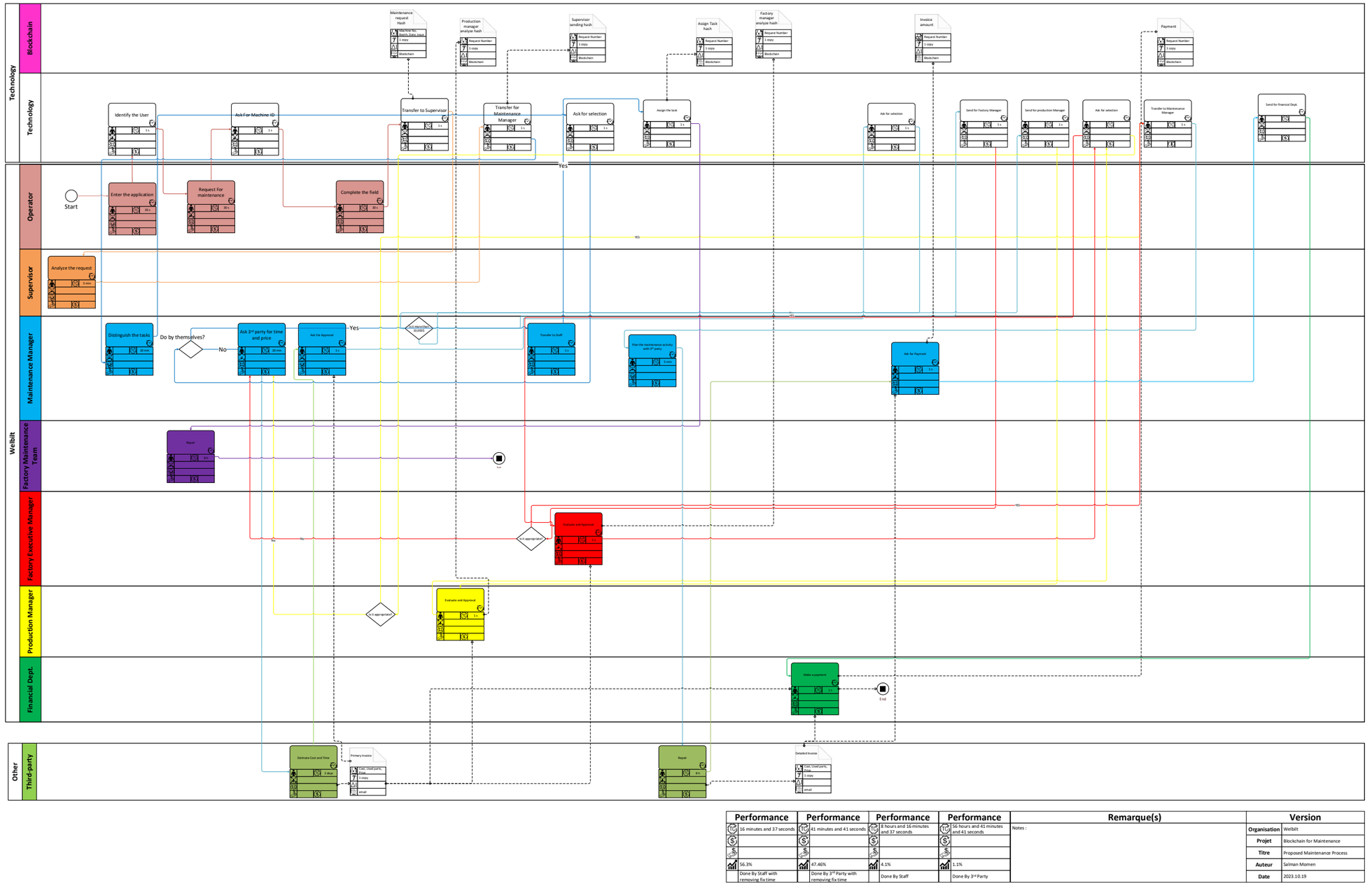
As part of the process enhancement, integrating technology into the maintenance workflow aims to improve efficiency and effectiveness. Building upon insights gained from mapping the existing maintenance process and defining the future flow, mapping the new process involves designing an updated and optimized workflow. This process includes integrating identified technological solutions to optimize various stages of maintenance management.

The mapping exercise visualizes revised steps, activities, and stakeholder roles, considering the utilization of technology at each stage. By mapping the new process, organizations establish a clear blueprint for the implementation of the optimized workflow, ensuring that the integration of technology aligns with the defined future flow. This mapping serves as a guiding framework for transitioning from the existing process to the improved maintenance system, enabling organizations to leverage technology for enhanced operational outcomes. This process is outlined using the GRMI4.0 proposed by Jeremie Mosser (Mosser, 2020).

Similar to the existing process, the operator also initiates the new process. Each operator has a user and an ID, which may be either his name or a unique number. The process is initiated when the operator logs in with a user ID, and the date and time are immediately recorded in the blockchain.

The operator is prompted to insert necessary information in the field, and upon form submission to the supervisor, the details are registered in the blockchain. The request is then forwarded to the supervisor, and all essential information is recorded in the blockchain. After verification by the supervisor, it is submitted and forwarded to the maintenance manager, generating a hash once again, indicating the time, date, and all other essential information completed by the operator and supervisor.

The process continues in alignment with the paper-based process, incorporating technology and blockchain. The complete process is illustrated in Figure 5.4 Future Process GRMI4.0 Diagram. Determining which part of the process will be registered on the blockchain and which will not be a crucial consideration.



Performance	Performance	Performance	Performance	Remarque(s)	Version	
16 minutes and 37 seconds	41 minutes and 41 seconds	8 hours and 16 minutes and 37 seconds	56 hours and 41 minutes and 41 seconds	Notes :	Organisation	Weibit
					Projet	Blockchain for Maintenance
					Titre	Proposed Maintenance Process
56.3%	47.46%	4.1%	1.1%		Auteur	Salman Momen
Done By Staff with removing fix time	Done By 3rd Party with removing fix time	Done By Staff	Done By 3rd Party		Date	2023.10.19

Figure 5.4 Future Process GRMI4.0 Diagram

5.2 Programming environment

For the creation of smart contracts and addressing maintenance issues, the Remix IDE was chosen as the primary development environment. The Remix IDE offers a user-friendly interface and a range of tools and features specifically designed for Ethereum smart contract development. Additionally, the integration of MetaMask, a popular browser extension wallet, was performed in the workflow. By connecting the Remix IDE with MetaMask, the virtual machine provided by MetaMask was leveraged for testing and interacting with the smart contracts. This combination allowed for effective development, testing, and validation of the smart contracts before their deployment to the Ethereum network. It should be noted that MetaMask itself interfaces with Ethereum's network, which runs on the Ethereum Virtual Machine (EVM), and does not provide a virtual machine or perform validation. Moreover, the deployment was conducted on the Ethereum Sepolia testnet network, ensuring the reliability and functionality of the smart contracts for maintenance purposes.

Remix IDE stands out for its user-friendly interface, making it easily accessible to developers. With its integrated development environment, developers can benefit from features such as syntax highlighting and auto-completion in the intuitive code editor, simplifying the process of writing and debugging smart contracts. Additionally, Remix IDE offers solid support for the Solidity language, including a built-in compiler that automates code compilation and bytecode generation, streamlining the overall development workflow.

Furthermore, Remix IDE seamlessly integrates with Metamask, a popular browser extension wallet, enhancing the capabilities of smart contract development. This integration allows developers to interact with smart contracts using a virtual machine provided by Metamask. Through this connection, developers can thoroughly test their smart contracts within a simulated Ethereum environment before deploying them to the actual network. By leveraging the virtual machine in Metamask, developers gain the ability to execute transactions, interact with contract functions, and observe real-time behavior of the contracts, ensuring their reliability and functionality. Moreover, Remix IDE offers plugin support, static analysis features, and is an open-source web-based platform, providing a comprehensive and customizable environment for smart contract development.

This integration between Remix IDE and Metamask offers several benefits. Firstly, it provides a convenient way to test and validate the functionality of smart contracts in a controlled environment. Developers can simulate different scenarios and inputs to ensure that the contract behaves as intended and handles various maintenance-related issues correctly.

Furthermore, Remix IDE's integration with Metamask simplifies the deployment process. Once the contract has been thoroughly tested in the virtual machine, developers can seamlessly deploy it to the Ethereum mainnet or any other test network supported by Metamask. This ensures a smooth transition from the development phase to the actual implementation of the contract for maintenance purposes.

In addition to the utilization of Remix IDE and Metamask, an interface was developed that establishes a connection with the blockchain using the Application Binary Interface (ABI). The ABI serves as an interface between the deployed smart contract on the blockchain and the front-end application. By leveraging the ABI, interaction with the smart contract functions and retrieval of relevant data from the blockchain were made possible. This interface seamlessly bridged the gap between the user interface and the underlying smart contract, facilitating efficient retrieval and display of maintenance-related information. With the presence of the ABI, users could easily access and interact with the smart contract through a user-friendly interface, enabling them to initiate maintenance requests, view status updates, and perform other relevant actions. The integration of the ABI further enhanced the usability and effectiveness of the maintenance system, providing users with a smooth and reliable experience while ensuring seamless connectivity with the deployed smart contracts on the blockchain.

The format and encoding rules for function calls and data retrieval from the smart contract are specified by the ABI. The function signatures, their input parameters, and the expected return types are outlined by it. By adhering to the ABI, accurate interaction with the smart contract's functions is ensured, with proper data exchange and execution.

Following the aforementioned overview, my thesis will delve into each aspect in detail. I will provide a comprehensive explanation of the smart contract creation process, including the compiling and testing stages. Additionally, I will explore the integration of Metamask and its usage

within the development workflow. The results obtained from the testing and deployment of the smart contract will be discussed, highlighting any significant findings. Furthermore, I will present an in-depth analysis of the interface developed, emphasizing the seamless connection achieved through the utilization of the Application Binary Interface (ABI). Each of these components will be examined independently, shedding light on their functionalities and contributions to the overall success of the maintenance system.

Sepolia is a test network (testnet) on the Ethereum blockchain. Testnets are alternative blockchain networks specifically designed for testing purposes. Deploying and testing smart contracts on a testnet allows developers to experiment with their code and identify potential bugs or vulnerabilities without incurring real gas fees and without risking any real funds.

When developers set up their Metamask wallet to connect to the Sepolia test network, they can obtain test Ether (ETH) from faucet services provided by the testnet. Test Ether is not real money but rather a simulation of Ethereum's native cryptocurrency. Developers can use this test Ether to deploy and execute transactions, including smart contract deployments and interactions, on the Sepolia testnet.

5.3 Implementation of smart contract and interface

Subsequently, a comprehensive test was conducted to validate the functionality of the implemented smart contract and interface. This testing phase aimed not only to ensure the proper operation of the technological components but also to measure the performance metrics of the enhanced maintenance process. By executing this test, we could ascertain the efficacy of the smart contract in automating and enforcing the proposed maintenance tasks. Additionally, the interface's seamless interaction with the blockchain and user-friendliness were evaluated. The test provided valuable insights into the overall performance of the blockchain-enhanced maintenance process, serving as a crucial step in validating the successful integration of technology into the established workflow.

As highlighted earlier, the development and deployment of the smart contract were undertaken within the Remix IDE environment, employing an unlicensed configuration and utilizing version 8.0 or newer iterations for the compilation phase Figure 5.5. Upon compilation for deployment and transaction execution, the connection to a compatible wallet was established, utilizing the "Walletconnect" option available in the environment field Figure 5.6.

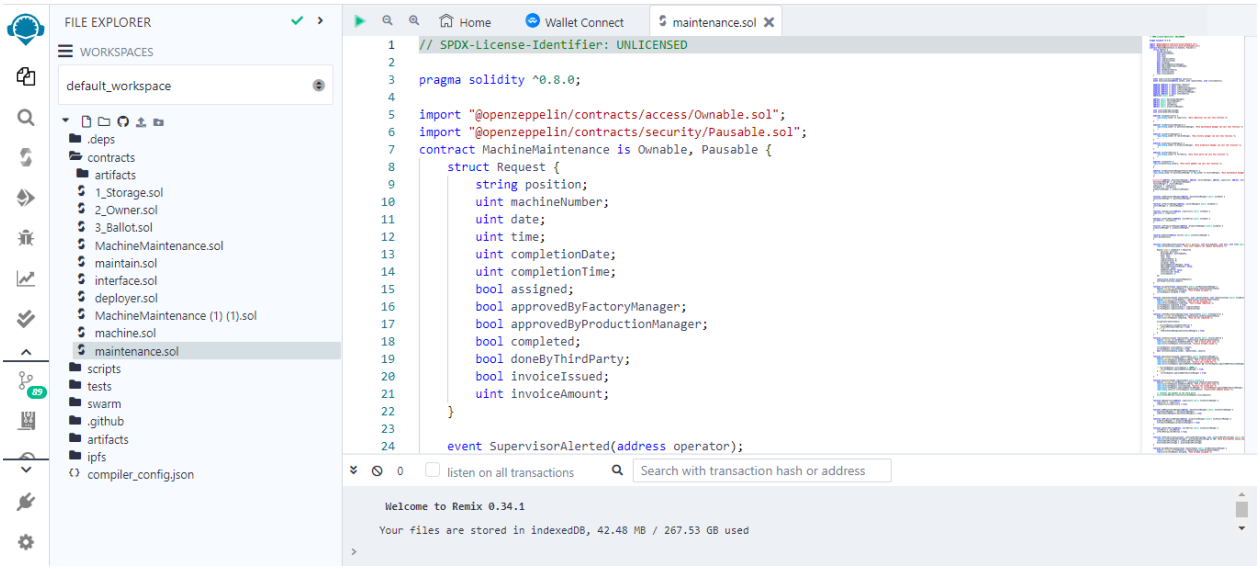


Figure 5.5 Contract in Remix

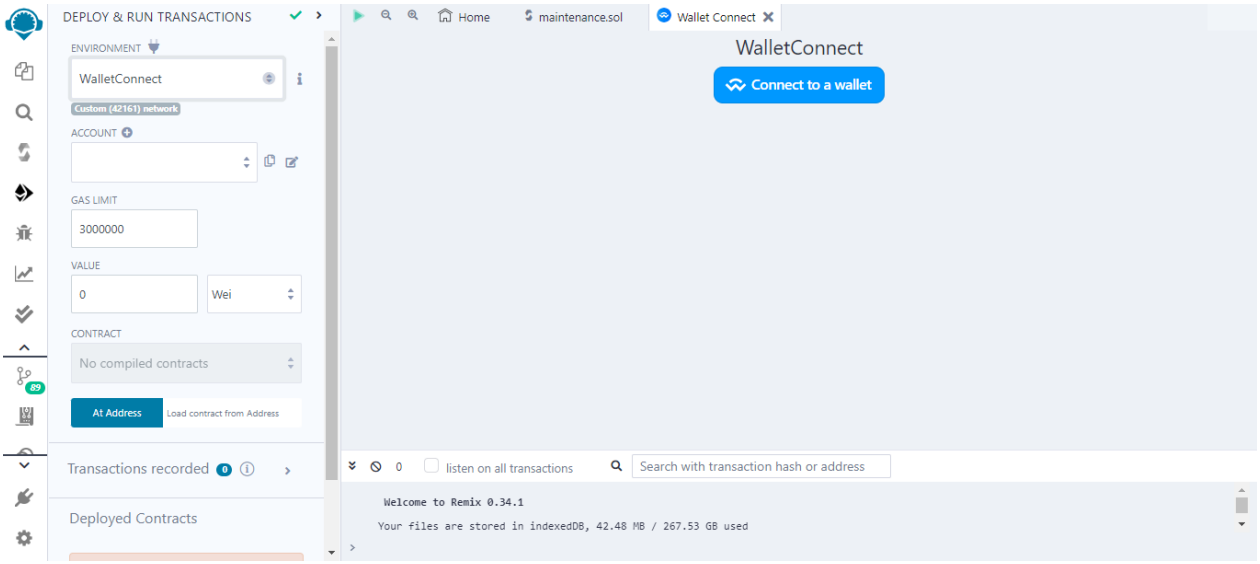


Figure 5.6 Compiling and connecting to Wallet

Subsequently, the testing environment was transitioned to the Metamask platform, followed by the establishment of a connection between the wallet and the designated Metamask environment, as illustrated in Figure 5.7.

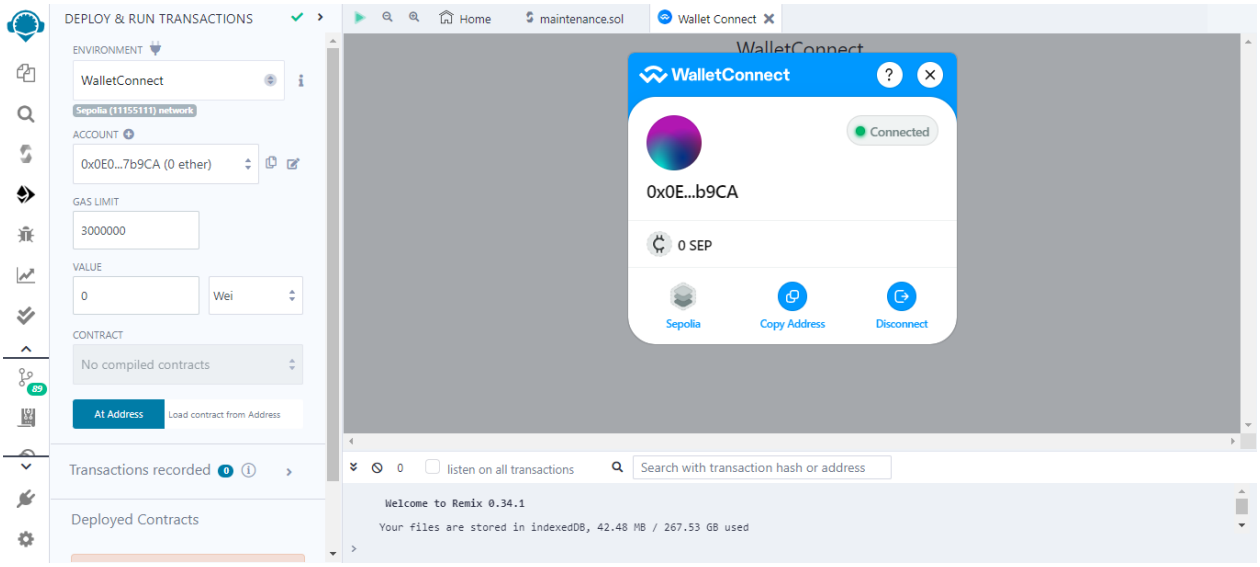


Figure 5.7 Wallet connect

Within the Metamask environment, a deliberate choice was made to leverage the Sepolia network for test execution. This decision was rooted in the need to replicate real-world conditions while maintaining a controlled testing environment Figure 5.8. Sepolia's network characteristics closely resemble the Ethereum mainnet, making it a suitable choice for accurate testing and validation of the smart contract's functionality, interactions, and responses. By selecting Sepolia for testing, the aim was to ensure that the smart contract's behavior and outcomes observed in the testing phase closely aligned with what could be anticipated in actual deployment scenarios on the Ethereum mainnet.

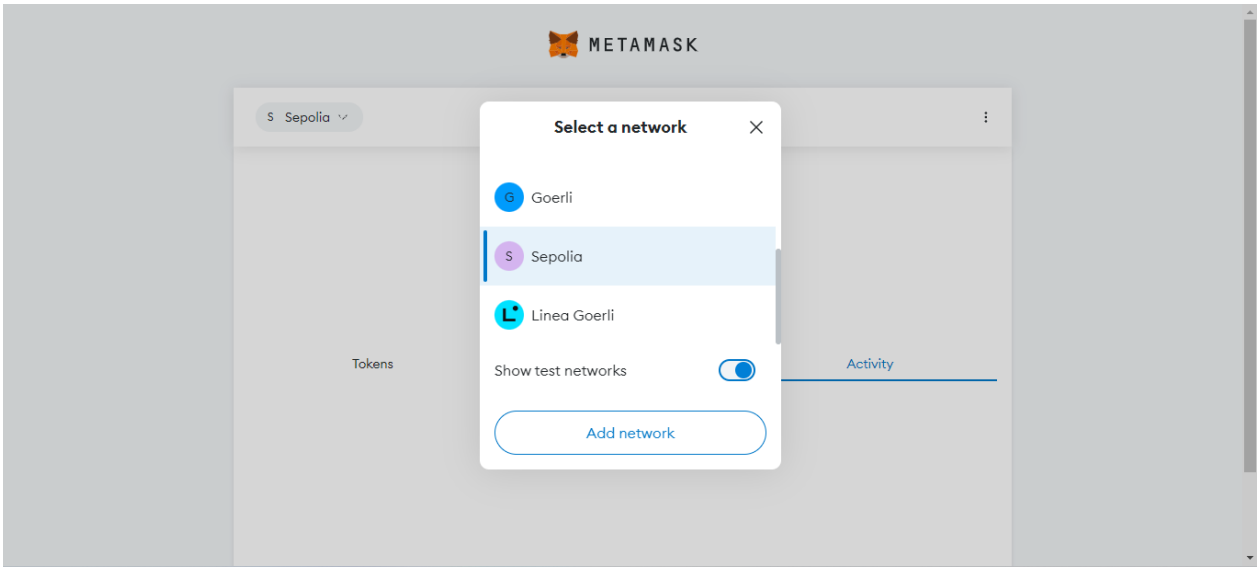


Figure 5.8 Selecting test environment in Metamask

The Sepolia web environment serves as a comprehensive interface through which the intricacies of the smart contract, including its functions and events, are made accessible for observation and analysis. This environment provides a real-time view into the smart contract's activities, enabling a clear understanding of how functions are executed, and events are triggered within the contract. By visualizing the contract's behavior and outcomes in this controlled yet dynamic environment, researchers and developers gain valuable insights into the contract's performance, its responsiveness to inputs, and the propagation of events. This transparency and observability are crucial for validating the contract's functionality, ensuring its alignment with the desired outcomes, and identifying any potential anomalies or discrepancies. In essence, the Sepolia web environment facilitates a holistic view of the smart contract's behavior, streamlining the evaluation process and contributing to the robustness of its design and execution, as illustrated in Figure 5.9.

Figure 5.9 Sepolia Environment

To illustrate the proof of concept, we initiated a maintenance request by interacting with the user interface. Through this interface, we seamlessly initiated the creation of a maintenance request, providing essential details such as the machine's designated position, marked as "Saw Machine," and assigning the machine number as "11222." This request was then initiated by submitting the query for maintenance. Subsequently, a distinct block, corresponding to this specific maintenance request, was generated, and recorded within the Blockchain, as visually depicted in Figure 5.10. This process showcases the practical realization of the smart contract's capabilities in action, underlining its ability to autonomously register and track transactions on the Blockchain. The seamless interaction between the user interface and the Blockchain solidifies the proof of concept, effectively demonstrating the smart contract's feasibility and its role in streamlining maintenance request management.

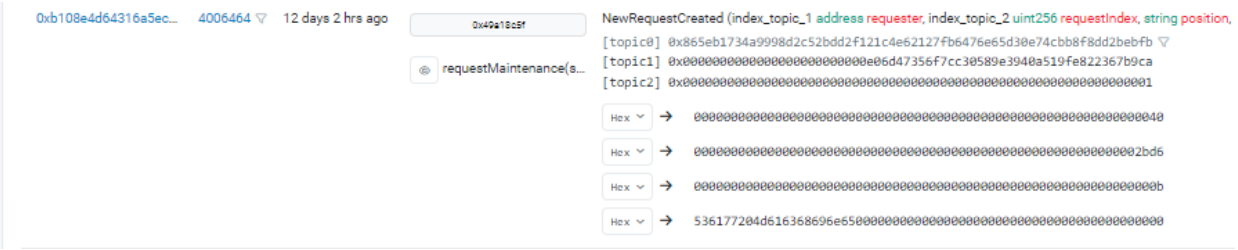


Figure 5.10 Block creating for request

You can see the detailed of the transactions if you click on each transaction. Figure 5.11.

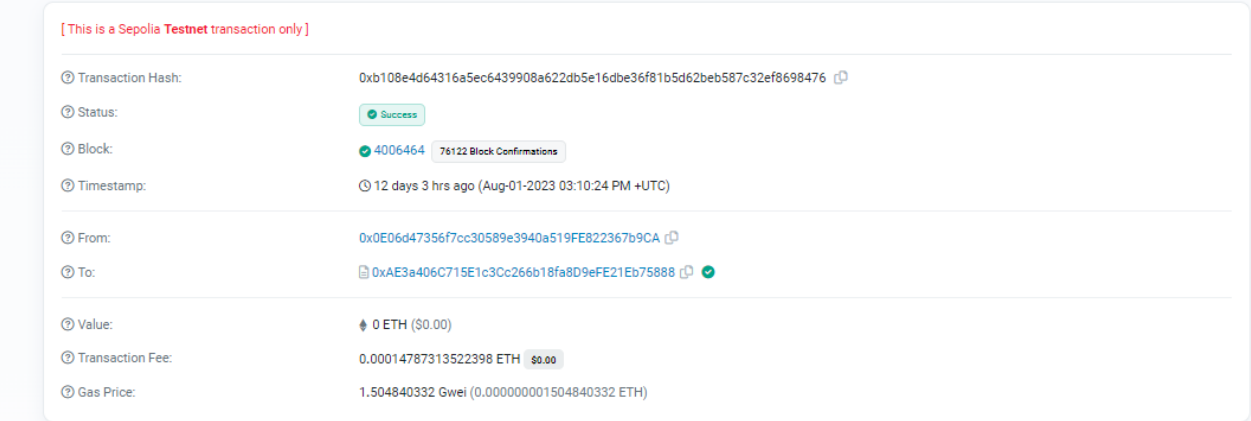


Figure 5.11 Block Detail

Upon the completion of each step within the main schedule, a consequential block is systematically generated within the blockchain. This newly formed block then undergoes the mining and consensus processes inherent to the blockchain network. As a result, the program meticulously compiles and furnishes a comprehensive list detailing the intricate specifications of the transaction. This encompassing list encapsulates pivotal information such as the transaction's associated hash value, the utilized gas amount, and other pertinent particulars. Importantly, embedded within this hash is an immutable timestamp, effectively documenting the precise date and time of the transaction's occurrence.

This meticulous recording mechanism within the blockchain ensures an indelible link between each step's execution and its corresponding timestamp. This feature empowers researchers, auditors, and stakeholders to comprehensively track and trace the sequence of events over time, thus fostering a robust framework for accountability and transparency. By leveraging the blockchain's inherent properties of data immutability and secure timestamping, this system provides an auditable trail of

events that safeguards the integrity of the process and facilitates comprehensive insights into the timing and execution of each step within the main schedule, as depicted in Figure 5.12.

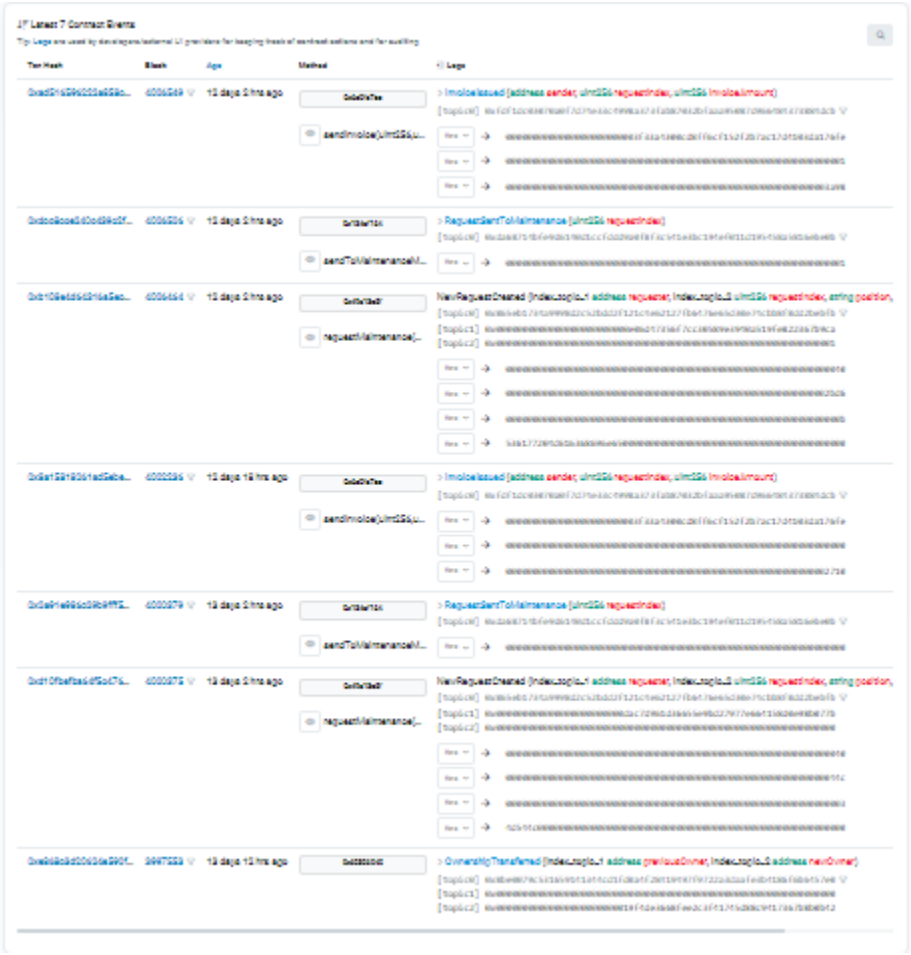


Figure 5.12 Blocks

In culmination, the demonstrated proof of concept, in conjunction with the meticulous testing conducted within the Metamask and Sepolia environments, underscores the effectiveness and viability of the developed smart contract. By seamlessly interfacing with the user interface and initiating maintenance requests, the smart contract showcases its capability to autonomously create, record, and manage tasks while ensuring a transparent and accountable workflow. The strategic utilization of the Metamask test environment, followed by the integration with the Sepolia network,

mirrors real-world conditions and confirms the contract's responsiveness and accurate replication of Ethereum mainnet interactions.

The smart contract's ability to consistently generate blocks within the blockchain, each timestamped with precise execution times, ensures an auditable and immutable record of events. The synergy between Metamask, Sepolia, and the smart contract underscores the contract's practicality, adaptability, and efficacy in managing maintenance requests. This successful proof of concept, validated through thorough testing, substantiates the smart contract's potential for revolutionizing maintenance workflows and bolstering the accountability and traceability of tasks.

5.4 Results

To facilitate our analysis, we categorized the maintenance steps into two distinct groups: those conducted by our company's staff and those done by third-party entities. Within each category, certain steps, such as repair activities and initial invoice preparations, were beyond our control. As a result, we segregated the time measurements into controlled time, including communication intervals, and uncontrolled time, where specific tasks were not under our influence.

The analysis of the paper-based maintenance process yields a detailed perspective on the time measures associated with each stage of the conventional workflow. The recorded data offers insights into the temporal aspects of the process, shedding light on the duration and sequences involved in the execution of maintenance tasks.

In the context of GRMI4.0, specific fields have been carefully chosen to gauge the performance of the maintenance process. As previously mentioned, we have segmented the process into two parts: tasks executed by the company staff and those delegated to third-party entities. Certain tasks fall into the category of uncontrollable variables, representing fixed-time elements, particularly those associated with maintenance actions or tasks outsourced to third-party companies. We will present two sets of comparative data, allowing us to identify and potentially eliminate these fixed-time elements. Our primary focus lies in the communication time, which serves as a variable time component, distinct from fixed elements. This emphasis on communication time contributes to a more dynamic evaluation, highlighting areas for improvement and efficiency gains within the process. The results shown in the figure 5.13.

Performance		Performance		Performance		Performance	
TC	38 minutes	TC	79 minutes and 20 second	TC	8 hours and 38 min	TC	57 hours and 19 min and 20 second
$		$		$		$	
$		$		$		$	
	N.A		N.A		N.A		N.A
	Done By Staff with Removing fix time		Done By 3rd Party with removing fix time		Done By Staff		Done By 3rd Party

Figure 5.13 Performance of Paper-Based Process

The performance and outcomes of the process, incorporating the new design with technology and blockchain integration, are illustrated in Figure 5.14.

Performance		Performance		Performance		Performance	
TC	16 minutes and 37 seconds	TC	41 minutes and 41 seconds	TC	8 hours and 16 minutes and 37 seconds	TC	56 hours and 41 minutes and 41 seconds
$		$		$		$	
$		$		$		$	
	56.3%		47.46%		4.1%		1.1%
	Done By Staff with removing fix time		Done By 3rd Party with removing fix time		Done By Staff		Done By 3rd Party

Figure 5.14 Performance of New Process

In the traditional third-party maintenance process, encompassing all steps, we observed a duration of 57 hours, 19 minutes, and 20 seconds. Through the implementation of blockchain technology, this time was significantly reduced to 56 hours, 41 minutes, and 41 seconds, representing a refinement of 1.1 percent. Moreover, by eliminating uncontrollable elements such as repair and invoice preparation times, and emphasizing communication intervals, the traditional time of 79 minutes and 20 seconds was reduced to 41 minutes and 41 seconds. This remarkable enhancement of 47.46 percent underscored the efficiency gains brought about by blockchain integration.

Similar improvements were observed in the processes managed by our company's staff. In the comprehensive traditional process, the duration was 8 hours and 38 minutes, which was fine-tuned

to 8 hours, 16 minutes, and 37 seconds with blockchain integration, resulting in a refinement of 4.1 percent. By excluding uncontrollable repair times and emphasizing communication efficiency, the original 38 minutes were reduced to 16 minutes and 37 seconds, reflecting a substantial 56.3 percent enhancement.

These significant time reductions not only demonstrate the tangible benefits of blockchain integration but also underscore the importance of streamlined communication and process optimization. Through our meticulous analysis, these findings provide empirical evidence of the transformative impact of blockchain technology on the maintenance processes, showcasing its potential to revolutionize industrial operations and improve overall efficiency within our organization.

CHAPTER 6 CONCLUSION AND RECOMMANDATIONS

6.1 Conclusion

In conclusion, our research delves into the transformative potential of blockchain technology in enhancing maintenance processes, with a specific focus on time efficiency. The results consistently highlight a substantial reduction in time and commendable improvements in operational efficiency across different facets of maintenance management. The adoption of blockchain in the traditional third-party maintenance process demonstrated a refined duration, achieving a 1.1 percent improvement. Notably, when emphasizing communication intervals and excluding uncontrollable elements, the time was further reduced by an impressive 47.46 percent and furthermore, our exploration into processes managed by our company's staff revealed noteworthy improvements, with a 4.1 percent refinement in the comprehensive traditional process duration. A closer look at communication efficiency and the exclusion of uncontrollable repair times resulted in a remarkable 56.3 percent enhancement. These findings substantiate the transformative impact of blockchain on maintenance operations.

Beyond these time-related metrics, the successful implementation of smart contracts, intricately designed based on a new process enriched with technological advancements, further validates our approach. The use of smart contracts automated and enforced key aspects of the newly modeled maintenance process. This implementation not only streamlined tasks and agreements related to maintenance operations but also provided a secure and efficient mechanism for executing these processes. The smart contract functionality, paired with a user-friendly interface, demonstrated the feasibility and effectiveness of integrating blockchain into maintenance management practices.

Acknowledging these findings, our research distinctively contributes to the field by providing empirical evidence of the transformative impact of blockchain on maintenance processes. The substantial time reductions validate the effectiveness of our proposed methodology, affirming that blockchain integration is a catalyst for performance improvements in maintenance operations.

In addition to the tangible efficiency gains highlighted in our research, it is imperative to underscore the newfound attributes of traceability and transparency that our enhanced maintenance process exhibits. The incorporation of blockchain technology inherently provides a traceable and transparent framework for maintenance operations. With every transaction and interaction immutably recorded in the blockchain ledger, stakeholders gain unparalleled visibility into the entire maintenance lifecycle. This transparency not only facilitates real-time monitoring but also ensures a comprehensive audit trail, enabling precise identification of each activity's origin and progression. The efficiency improvements achieved through blockchain integration are now complemented by an inherent traceability mechanism, enhancing accountability and governance in maintenance practices within the Industry 4.0 landscape.

In summary, our comprehensive research, backed by detailed validation, provides empirical evidence of the multifaceted benefits of blockchain integration in maintenance processes. The success of smart contract implementation and the intuitive user interface further reinforce the practicality and viability of blockchain solutions in revolutionizing maintenance management within the Industry 4.0 context. Our findings contribute not only to academic understanding but also offer valuable insights for industry practitioners seeking tangible improvements in maintenance efficiency.

6.2 Recommendations

Prioritize training programs to equip staff with blockchain skills for effective system implementation and continuous improvement. Encourage collaboration between academia and industry to drive innovation in tailored blockchain applications for diverse industrial sectors. Collaborate on industry-wide standards for blockchain implementation to enhance interoperability and applicability across diverse contexts. Establish mechanisms for continuous monitoring and evaluation of maintenance processes to ensure ongoing optimization. Explore integration with emerging technologies like IoT and AI to create a powerful synergy for proactive maintenance strategies.

These concise recommendations could guide organizations in harnessing blockchain's potential for enhanced operational efficiency and sustainable maintenance practices in Industry 4.0.

6.3 Limitations

The integration of blockchain in maintenance management, while promising, encounters technical challenges. Scalability issues and high transaction costs on certain blockchain platforms may pose difficulties, especially when handling extensive maintenance-related data. This limitation restricts the application of blockchain solutions in scenarios requiring significant data processing.

Integrating blockchain with existing legacy systems and databases, as explored in the thesis, proves to be complex. Achieving seamless data interoperability demands careful planning and execution, requiring substantial resources and expertise. The intricacies of migrating historical maintenance data to blockchain structures are pertinent limitations.

Environmental considerations in the specific context of maintenance operations and blockchain implementation are highlighted. The energy-intensive nature of certain blockchain mechanisms, such as Proof of Work (PoW), raises ethical concerns that need careful consideration.

Acknowledging the specific applications explored in the thesis, the dependence on network connectivity for blockchain-based maintenance operations is recognized as a potential limitation. This dependency may pose challenges in scenarios with limited internet access, impacting real-time data synchronization.

6.4 Future research directions

Future research should prioritize scalability in blockchain-driven maintenance management. Investigating advanced consensus algorithms and off-chain solutions is crucial. Interdisciplinary efforts are needed for standardized protocols, enabling seamless integration between diverse blockchain platforms and maintenance systems.

Exploring the integration of blockchain with IoT and AI remains a key focus. Research should delve into how AI algorithms leverage blockchain-stored data for predictive maintenance insights.

Developing real-time analytics solutions for blockchain networks is vital for efficient maintenance processes. Exploring how blockchain enables instant analysis and proactive responses to equipment failures through predictive maintenance algorithms is essential.

Understanding human factors in blockchain adoption in maintenance contexts is crucial. Exploring user experiences and attitudes toward blockchain interfaces can enhance user acceptance in industrial maintenance settings.

Sustainable blockchain practices could be a core focus for future research. Exploring energy-efficient mechanisms and assessing the environmental impact of blockchain networks are essential steps.

REFERENCES

Abbas, Y., Martinetti, A., Moerman, J.-J., Hamberg, T., & van Dongen, L. A. M. (2020). Do you have confidence in how your rolling stock has been maintained? A blockchain-led knowledge-sharing platform for building trust between stakeholders. *International Journal of Information Management, 55*. https://doi.org/10.1016/j.ijinfomgt.2020.102228

Basri, E. I., Abdul Razak, I. H., Abdul Samat, H., & Kamaruddin, S. (2017, 05/08). Preventive Maintenance (PM) planning: a review. *Journal of Quality in Maintenance Engineering, 23*. https://doi.org/10.1108/JQME-04-2016-0014

Bhushan, B., Sinha, P., Sagayam, K. M., & J, A. (2021, 2021/03/01/). Untangling blockchain technology: A survey on state of the art, security threats, privacy services, applications and future research directions. *Computers & Electrical Engineering, 90*, 106897. https://doi.org/https://doi.org/10.1016/j.compeleceng.2020.106897

Chang, F., Zhou, G., Zhang, C., Ding, K., Cheng, W., & Chang, F. (2021). A maintenance decision-making oriented collaborative cross-organization knowledge sharing blockchain network for complex multi-component systems. *Journal of Cleaner Production, 282*. https://doi.org/10.1016/j.jclepro.2020.124541

Danjou, C., Rivest, L., & Pellerin, R. (2017). *PME 2.0 : Le passage au numérique : Industrie 4.0 : des pistes pour aborder l'ère du numérique et de la connectivité* [Technical Report]. https://espace2.etsmtl.ca/id/eprint/14934/

Drath, R., & Horch, A. (2014). Industrie 4.0: Hit or Hype? [Industry Forum]. *IEEE Industrial Electronics Magazine, 8*(2), 56-58. https://doi.org/10.1109/MIE.2014.2312079

Erboz, G. (2017). *How To Define Industry 4.0: Main Pillars Of Industry 4.0*. Managerial Trends in the Development of Enterprises in Globalization Era, Nitra, 761-767.

Kasinathan, P., Martintoni, D., Hofmann, B., Senni, V., & Wimmer, M. (2021). Secure remote maintenance via workflow-driven security framework.*2021 IEEE International Conference on Blockchain (Blockchain)* 2021 IEEE International Conference on Blockchain (Blockchain), 6-8 Dec. 2021, Piscataway, NJ, USA.

Komalavalli, C., Saxena, D., & Laroiya, C. (2020). Chapter 14 - Overview of Blockchain Technology Concepts. In S. Krishnan, V. E. Balas, E. G. Julie, Y. H. Robinson, S. Balaji, & R. Kumar (Eds.), *Handbook of Research on Blockchain Technology* (pp. 349-371). Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-12-819816-2.00014-9

Lee, J., Bagheri, B., & Kao, H.-A. (2015, 2015/01/01/). A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing Letters, 3*, 18-23. https://doi.org/https://doi.org/10.1016/j.mfglet.2014.12.001

Li, B., Mi, H., Min, L., & Jingwei, W. (2019, /). BPIIoT: A Light-Weighted Blockchain-Based Platform for Industrial IoT. *IEEE Access, 7*, 58381-58393. https://doi.org/10.1109/ACCESS.2019.2914223 (IEEE Access (USA))

Mosser, J. (2020). *Cartographie 4.0 pour la transformation numérique des processus* [Master's thesis, Polytechnique Montréal]. https://publications.polymtl.ca/5379/

Moubray, J. (2001). *Reliability-centered Maintenance*. Industrial Press. https://books.google.ca/books?id=bNCVF0B7vpIC

Nakamoto, S. (2009, 03/24). Bitcoin: A Peer-to-Peer Electronic Cash System. *Cryptography Mailing list at https://metzdowd.com*.

Ownable. (2023). *openzeppelin docs Ownable Liberary*. https://docs.openzeppelin.com/contracts/4.x/access-control#ownable.

Patidar, L., Soni, V. K., & Soni, P. K. (2017). Maintenance strategies and their combine impact on manufacturing performance. *International Journal of Mechanical and Production Engineering Research and Development, 7*(1), 13-22.

Pausable. (2023). *openzeppelin docs Pausable Liberary*

https://docs.openzeppelin.com/contracts/4.x/api/utils#Pausable.

Qiuan, C., Zhenwei, Z., Shubin, S., & Zhiqiang, C. (2020). Intelligent Maintenance of Complex Equipment Based on Blockchain and Digital Twin Technologies.*2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* 2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 14-17 Dec. 2020, Piscataway, NJ, USA.

Sang, G. M., Xu, L., Vrieze, P. d., Bai, Y., & Pan, F. (2021). *Predictive Maintenance in Industry 4.0* Proceedings of the 10th International Conference on Information Systems and Technologies, Lecce, Italy. https://doi.org/10.1145/3447568.3448537

Stodt, J., Jastremskoj, E., Reich, C., Welte, D., & Sikora, A. (2019). Formal Description of Use Cases for Industry 4.0 Maintenance Processes Using Blockchain Technology.*2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing*

Systems: Technology and Applications (IDAACS). Proceedings 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 18-21 Sept. 2019, Piscataway, NJ, USA.

Tripathi, G., Ahad, M. A., & Casalino, G. (2023, 2023/12/01/). A comprehensive review of blockchain technology: Underlying principles and historical background with future challenges. *Decision Analytics Journal, 9*, 100344. https://doi.org/https://doi.org/10.1016/j.dajour.2023.100344

Welte, D., Sikora, A., SchOnle, D., Stodt, J., & Reich, C. (2020). Blockchain at the Shop Floor for Maintenance.*2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)* 2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 29-30 Oct. 2020, Piscataway, NJ, USA.

Wichmann, R., Eisenbart, B., & Gericke, K. (2019, 07/01). The Direction of Industry: A Literature Review on Industry 4.0. *1*, 2129-2138. https://doi.org/10.1017/dsi.2019.219

Yang, F., & Gu, S. (2021, 2021/06/01). Industry 4.0, a revolution that requires technology and national strategies. *Complex & Intelligent Systems, 7*(3), 1311-1325. https://doi.org/10.1007/s40747-020-00267-9

Zheng, Z., Xie, S., Dai, H.-N., Chen, X., & Wang, H. (2017). *An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends*. https://doi.org/10.1109/BigDataCongress.2017.85

Zonta, T., da Costa, C. A., da Rosa Righi, R., de Lima, M. J., da Trindade, E. S., & Li, G. P. (2020, 2020/12/01/). Predictive maintenance in the Industry 4.0: A systematic literature review. *Computers & Industrial Engineering, 150*, 106889. https://doi.org/https://doi.org/10.1016/j.cie.2020.106889

APPENDIX A SMART CONTRACT

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/security/Pausable.sol";

contract MachineMaintenancetest6 is Ownable, Pausable {
   struct Request {
    string position;
    uint machineNumber;
    bool assigned;
    bool approvedByFactoryManager;
    bool approvedByProductionManager;
    bool completed;
    bool doneByThirdParty;
    bool invoiceIssued;
    uint invoiceAmount;
    bool senttoMaintenanceManager;
    uint internalWorkPercentage;
    uint externalWorkPercentage;
    }

    // Array to store all requests
    Request[] public allRequests;

    event SupervisorAlerted(address operator);
    event InvoiceIssued(address sender, uint requestIndex, uint invoiceAmount);
    event NewRequestCreated(address indexed requester, uint indexed requestIndex,
string position, uint machineNumber);
    event RequestSentToMaintenance(uint requestIndex);

    mapping (address => Request[]) requests;
    mapping (address => bool) isSupervisor;
    mapping (address => bool) isMaintenanceManager;
    mapping (address => bool) isFactoryManager;
    mapping (address => bool) isProductionManager;
    mapping (address => bool) isThirdParty;
    address[] public staff;

    address public maintenanceManager;
    address public factoryManager;
```

```
    address public supervisor;
    address public thirdParty;
    address public productionManager;

    uint internalWorkPercentage;
    uint externalWorkPercentage;
    uint internalPercentage = getInternalWorkPercentage();
    uint externalPercentage = getExternalWorkPercentage();

    modifier onlySupervisor() {
        require(msg.sender == supervisor, "Only supervisor can call this
function.");
        _;
    }

    modifier onlyMaintenanceManager() {
        require(msg.sender == maintenanceManager, "Only maintenance manager can
call this function.");
        _;
    }

    modifier onlyFactoryManager() {
        require(msg.sender == factoryManager, "Only factory manager can call this
function.");
        _;
    }

    modifier onlyProductionManager() {
        require(msg.sender == productionManager, "Only production manager can
call this function.");
        _;
    }

    modifier onlyThirdParty() {
        require(msg.sender == thirdParty, "Only third party can call this
function.");
        _;
    }

    modifier onlyStaff() {
    require(isStaff(msg.sender), "Only staff members can call this function.");
    _;
    }

    modifier onlyMaintenanceManagerOrFactoryManager() {
```

```
    require(msg.sender == maintenanceManager || msg.sender == factoryManager,
"Only maintenance manager or factory manager can call this function.");
    _;
    }

   constructor(address _maintenanceManager, address _factoryManager, address
_supervisor, address _thirdParty, address _productionManager) {
    maintenanceManager = _maintenanceManager;
    factoryManager = _factoryManager;
    supervisor = _supervisor;
    thirdParty = _thirdParty;
    productionManager = _productionManager;
    }

    function getInternalWorkPercentage() public view returns (uint) {
    return internalWorkPercentage;
    }

    function getExternalWorkPercentage() public view returns (uint) {
    return externalWorkPercentage;
    }

    function setMaintenanceManager(address _maintenanceManager) public onlyOwner
{
    maintenanceManager = _maintenanceManager;
    }

    function setFactoryManager(address _factoryManager) public onlyOwner {
    factoryManager = _factoryManager;
    }

    function setSupervisor(address _supervisor) public onlyOwner {
    supervisor = _supervisor;
    }

    function setThirdParty(address _thirdParty) public onlyOwner {
    thirdParty = _thirdParty;
    }

    function setProductionManager(address _productionManager) public onlyOwner {
    productionManager = _productionManager;
    }


    function addStaff(address wallet) public onlyFactoryManager {
```

```
    staff.push(wallet);
    }


  function requestMaintenance(string memory position, uint machineNumber) public
onlyStaff returns (uint) {
    require(isStaff(msg.sender), "Only staff members can request maintenance.");

    Request memory newRequest = Request({
        position: position,
        machineNumber: machineNumber,
        senttoMaintenanceManager: false,
        assigned: false,
        approvedByFactoryManager: false,
        approvedByProductionManager: false,
        completed: false,
        doneByThirdParty: false,
        invoiceIssued: false,
        invoiceAmount: 0,
        internalWorkPercentage: 0,
        externalWorkPercentage: 0
    });

     allRequests.push(newRequest);
    uint newIndex = allRequests.length - 1; // Get the index of the newly created
request
    emit NewRequestCreated(msg.sender, newIndex, position, machineNumber); //
Emit the event with the request number

    return newIndex; // Return the index of the newly created request
    }

    //Function to retrieve request data by its number
   function getRequestData(uint requestIndex) public view returns (string memory,
uint, bool, bool, bool, bool, bool, bool, uint) {
    require(requestIndex < allRequests.length, "Invalid request index");

    Request storage currentRequest = allRequests[requestIndex];
    return (
        currentRequest.position,
        currentRequest.machineNumber,
        currentRequest.assigned,
        currentRequest.approvedByFactoryManager,
        currentRequest.approvedByProductionManager,
        currentRequest.completed,
```

```
        currentRequest.doneByThirdParty,
        currentRequest.invoiceIssued,
        currentRequest.invoiceAmount
    );
    }

    function setWorkDistribution(uint requestIndex, uint internalPercentage, uint
externalPercentage) public onlyMaintenanceManager {
    require(requestIndex < allRequests.length, "Invalid request index");
    require(internalPercentage + externalPercentage == 100, "Work distribution
should sum up to 100.");

    allRequests[requestIndex].internalWorkPercentage = internalPercentage;
    allRequests[requestIndex].externalWorkPercentage = externalPercentage;
    }


    function decideWorkAssignment(uint requestIndex) public
onlyMaintenanceManager {
    Request storage currentRequest = allRequests[requestIndex];
    require(!currentRequest.assigned, "Task already assigned.");

    if (currentRequest.externalWorkPercentage > 0) {
        currentRequest.assigned = true;
        currentRequest.doneByThirdParty = true;
    } else {
        currentRequest.assigned = true;
        currentRequest.doneByThirdParty = false;
    }
    }


    function completeTask(uint requestIndex) public
onlyMaintenanceManagerOrFactoryManager {
    Request storage currentRequest = allRequests[requestIndex];
    require(currentRequest.assigned, "Task not yet assigned.");
    require(!currentRequest.completed, "Task already completed.");
    currentRequest.completed = true;
    }


    function sendToMaintenanceManager(uint requestIndex) public onlySupervisor {
        require(requestIndex < allRequests.length, "Invalid request index");

        // Mark the request as sent to maintenance manager
        allRequests[requestIndex].senttoMaintenanceManager = true;
```

```
        emit RequestSentToMaintenance(requestIndex);
    }

   function sendInvoice(uint requestIndex, uint amount) public onlyThirdParty {
    require(requestIndex < allRequests.length, "Invalid request index");

    Request storage currentRequest = allRequests[requestIndex];
    require(currentRequest.doneByThirdParty, "Not a third party task.");
    require(!currentRequest.invoiceIssued, "Invoice already issued.");

    currentRequest.invoiceAmount = amount;
    currentRequest.invoiceIssued = true;
    emit InvoiceIssued(msg.sender, requestIndex, amount);
}


    function approveInvoice(uint requestIndex) public onlyFactoryManager {
    require(requestIndex < allRequests.length, "Invalid request index");

    Request storage currentRequest = allRequests[requestIndex];
    require(currentRequest.doneByThirdParty, "Not a third party task.");
    require(currentRequest.invoiceIssued, "Invoice not issued yet.");
    require(!currentRequest.approvedByFactoryManager &&
!currentRequest.approvedByProductionManager, "Invoice already approved.");

    if (currentRequest.invoiceAmount > 10000) {
        currentRequest.approvedByFactoryManager = true;
    } else {
        currentRequest.approvedByProductionManager = true;
    }
}


    function payInvoice(uint requestIndex) public payable onlyOwner {
        Request storage currentRequest = allRequests[requestIndex];
        require(currentRequest.doneByThirdParty, "Not a third party task.");
        require(currentRequest.invoiceIssued, "Invoice not issued yet.");
        require(currentRequest.approvedByFactoryManager ||
currentRequest.approvedByProductionManager, "Invoice not approved.");
        require(msg.value >= currentRequest.invoiceAmount, "Insufficient payment
amount.");

        // Transfer the payment to the third party
        payable(thirdParty).transfer(currentRequest.invoiceAmount);
```

```
    }

    function addSupervisor(address _supervisor) public onlyFactoryManager {
        supervisor = _supervisor;
        isSupervisor[_supervisor] = true;
    }

    function addMaintenanceManager(address _maintenanceManager) public
onlyFactoryManager {
        maintenanceManager = _maintenanceManager;
        isMaintenanceManager[_maintenanceManager] = true;
    }

    function addProductionManager(address _productionManager) public
onlyFactoryManager {
        productionManager = _productionManager;
        isProductionManager[_productionManager] = true;
    }

    function addThirdParty(address _thirdParty) public onlyFactoryManager {
        thirdParty = _thirdParty;
        isThirdParty[_thirdParty] = true;
    }




    function alertSupervisor(address operator) internal {
        require(isSupervisor[supervisor], "Supervisor not set.");
        // Additional logic to alert the supervisor could be done outside the
contract (just a msg.json file as use case)

        emit SupervisorAlerted(operator);
    }

    function isStaff(address wallet) public view returns (bool) {
    for (uint i = 0; i < staff.length; i++) {
        if (staff[i] == wallet) {
            return true;
        }
    }
    return false;
    }
```

```
    function removeStaff(address wallet) public onlyFactoryManager {
    for (uint i = 0; i < staff.length; i++) {
        if (staff[i] == wallet) {
            // Remove the staff member from the array by shifting elements
            for (uint j = i; j < staff.length - 1; j++) {
                staff[j] = staff[j + 1];
            }
            // Reduce the length of the array by 1
            staff.pop();
            return;
        }
    }
    }


}
```