

Blockchain for Smart Manufacturing Enterprises

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Business Informatics

eingereicht von

Gregor Liebenberger

Matrikelnummer 01152696

an der

Fakultät für Informatik der Technischen Universität Wien

Betreuung

Betreuer (1): Univ.-Prof. Dr.-Ing Wilfried Sihn

Betreuer (2): Dr.-Ing., Assistant Professor Fazel Ansari

Wien, TT.MM.JJJJ

(Unterschrift Verfasser)

(Unterschrift Betreuer)



TECHNISCHE
UNIVERSITÄT
WIEN

„Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst, die verwendeten Quellen und Hilfsmittel vollständig angegeben und Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.“

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.



Wien, im September 2019

Gregor Liebenberger

Acknowledgments

Firstly, I would like to thank my supervisor Professor Fazel Ansari for his continued support and guidance throughout the writing of this thesis. I am grateful for the detailed feedback provided by him with the goal of improving quality.

Secondly, I have to give a big thank you to TU Wien, who made the writing of this thesis possible. Through my study of Business Informatics, I feel comfortable to work in this field.

Thirdly and mostly I must thank my family, in particular my parents and grandparents, who supported me heavily and pushed me in times of need. Their support was not only financially, but also by giving guidance and patience when needed.

Lastly, I have to thank my girlfriend, who never stopped reading this thesis and finding typing errors as well as giving valuable input.

Gregor Liebenberger

Kurzfassung

Anwendungsfälle für Blockchain beschränken sich schon lange nicht mehr nur auf Kryptowährungen. Durch ständige Weiterentwicklung steigt das Potential von Blockchain stetig. Vor allem durch die Einführung von Smart Contracts, haben sich die Möglichkeiten Blockchain einzusetzen vervielfältigt. In Kombination mit starken kryptografischen Mechanismen kann Blockchain-Technologie die Sicherheit erhöhen und aufgrund von Dezentralität auch Probleme umgehen, welche durch Zentralisierung entstehen.

Smart Manufacturing und Industrie 4.0 kombinieren verschiedene Technologien wie das Internet der Dinge, Cloud Computing, Smarte Produktionsstätten und vieles mehr. Mit dieser riesigen Sammlung unterschiedlicher Technologien ergeben sich allerdings weitere Herausforderungen. Diese Arbeit diskutiert Möglichkeiten, Blockchain in der intelligenten Fertigung einzusetzen und zeigt verschiedene Anwendungsfälle auf. Ein interessanter Anwendungsfall ist beispielsweise die Zusammenarbeit zwischen Geschäftspartnern. Der sichere Datenaustausch dabei ist eine große Herausforderung bei der Implementierung von Wartungsstrategien.

Blockchain bietet großartige Mechanismen zur Lösung der Probleme, die beim Datenaustausch auftreten. Die beiden wichtigsten Sicherheitsmerkmale für die gemeinsame Nutzung von Daten sind das Ausblenden von Informationen und die Authentifizierung von Informationen. Der in dieser Arbeit vorgestellte Prototyp nimmt sich diesem Problem an. Da eine private Ethereum-Blockchain verwendet wird, haben nur autorisierte Benutzer Zugriff auf die in der Blockchain gespeicherten Daten. Darüber hinaus wird durch die Unveränderlichkeit der Daten sowie das Nachvollziehen von Transaktionen eine hohe Transparenz erreicht.

Zusammenfassend kann Blockchain einige Herausforderungen in der intelligenten Fertigung lösen, aber konkrete Anwendungsfälle müssen sorgfältig abgewogen werden. Eine umfangreiche Analyse ist erforderlich, um festzustellen, ob eine Blockchain die beste Lösung ist, oder ob andere Technologien für ein bestimmtes Szenario besser geeignet sind.

Abstract

Blockchain is attracting more and more attention not only in the context of cryptocurrencies but also for a wide variety of other use cases. Through further development the potential of blockchain expanded. A key factor of this development was the introduction of smart contracts. In combination with strong cryptographical mechanisms, blockchain technology can increase security and, due to its decentralized nature, bypass problems arising from centralization.

Smart manufacturing combines several technologies, such as the Internet of Things, cloud computing, smart factories and many others. Through the integration of these different technologies, several challenges appear. A prominent example is cybersecurity. This thesis discusses the opportunities blockchain provides in smart manufacturing and will show several use cases. For example, one critical use case is collaborative work among business partners. Indeed, securely sharing data is a huge challenge when implementing maintenance strategies in smart manufacturing enterprises.

Blockchain provides effective mechanisms to solve the issues caused by data sharing such as data immutability, timestamps and traceability to name a few. The two main security characteristics for data sharing are information hiding and information authentication. The prototype introduced in this work contributes to solving these issues. Since a private Ethereum blockchain has been used, only authorized users have obtained access to stored data on the blockchain. Additionally, high transparency provides data immutability and the tracing of transactions.

In conclusion, blockchain can solve challenges such as secure data collection, resistance against cyber-attacks, trust among partners and more due to decentralized validation, data immutability and transparency. However, the use case must be evaluated carefully. An in-depth analysis is necessary to distinguish whether a blockchain is the optimal (best-fitting) solution or whether other technologies fulfil the requirements better.

Contents

1	Introduction	4
1.1	Motivation and Focus of Research	4
1.2	Problem Definition and Research Questions	5
1.3	Solution Approach	7
1.4	Structure of the Thesis	9
2	Theoretical Foundations.....	11
2.1	Smart Manufacturing: History & Terminologies	11
2.1.1	Challenges	17
2.1.2	Limitations in Manufacturing	19
2.1.3	Advantages & Disadvantages	20
2.2	Blockchain	22
2.2.1	History & Applications	22
2.2.2	Definition	24
2.2.3	Consensus algorithms	26
2.2.4	Blockchain implementations	27
2.2.5	Challenges	28
2.2.6	Limitations.....	29
2.2.7	Advantages & Disadvantages	30
2.3	Blockchain in Smart Manufacturing	32
2.3.1	Challenges	35
2.3.2	Advantages & Disadvantages	37
2.3.3	Technological and Non-Technological Risks	39
2.3.4	Use-cases	42
3	Security in Manufacturing.....	44
3.1	Cybersecurity in Industry 4.0	44
3.1.1	Encryption of Computer Aided Design Models.....	48
3.1.2	Cyberphysical Security	49
3.1.3	Threats.....	52
3.1.4	SCADA Systems.....	53
3.1.5	Security in IoT	56

3.2	Similar solutions for the problem in maintenance	59
3.3	Blockchain solutions in other sectors.....	61
4	Blockchain Collaboration Model.....	64
4.1	Concept and Specification.....	66
4.1.1	Blockchain System.....	66
4.1.2	Functional Requirements	68
4.1.3	Non-Functional Requirements	72
4.2	Design	73
4.2.1	Architecture.....	73
4.2.2	Data Model.....	74
4.3	Implementation.....	75
4.3.1	Configuration.....	75
4.3.2	Smart Contracts	76
4.3.3	Front-End.....	77
4.3.4	Evaluation and Validation	83
5	Discussion & outlook.....	89
5.1	Discussion	89
5.2	Outlook.....	91
6	Bibliography	93
7	Attachment.....	100
7.1	Smart contract	100
7.1.1	Machine	100
7.1.2	Ticket	104
7.1.3	Spare Part.....	106
7.1.4	Spare Part Request	107
7.2	Front end.....	108
7.2.1	Creating a Spare Part	108
7.2.2	Viewing a Spare Part	111
7.3	Installation Guide.....	112
8	List of figures.....	117
9	List of tables.....	119
10	List of Code Fragments	120

11 List of abbreviations121

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.



1 Introduction

This introductory section provides an overview about the entire thesis, including motivation, problem statements, research objectives, methodology of research as well as the structure of this thesis.

1.1 Motivation and Focus of Research

Information and communication technology (ICT) are developing rapidly supported by several data-driven and smart technologies such as the Internet of Things (IoT), Artificial Intelligence (AI), Big Data and Cloud Computing (Rittinghouse and Ransome 2016). So-called cyber physical systems (CPS) enable merging of the virtual and physical world. The introduction as well as integration of CPS into the industry sector marks the start of the fourth industrial revolution and Industry 4.0 (Zheng et al. 2018, p. 137–138).

A critical goal in Industry 4.0 is to collect as much data as possible from different components distributed all over a factory to gain a deeper insight into the production process (Fernández-Caramés and Fraga-Lamas 2019, p. 1). With this objective in mind, connectivity plays an important role. IoT, or more precisely in the context of smart manufacturing Industrial IoT, is a great tool for connectivity due to its great sensing and communication technology. Therefore, a change from a central server-based approach is needed, where ideally all components should be able to communicate with each other. Figure 1 shows a standard approach with one central control unit on the left side and a distributed approach based on a peer-to-peer network on the right side (Park et al. 2018, p. 9–11).

A peer-to-peer network relies on the concept that every peer is equal and can appropriate information and communication. No central coordination is needed if two peers spontaneously decide to communicate with each other (Schoder and Fischbach 2003, p. 27–28). The difference to a centralized traditional model is the promise of scalability and lower cost of ownership in combination with a self-organized organization (Schoder et al. 2004, p. 2). A peer-to-peer network can be divided into three levels: communication level, an applications level and the infrastructure level (Schoder et al. 2004, p. 3–5). Blockchain is an example of a technology using a peer-to-peer network with each node being one peer in the network. (Bahga and Madisetti 2016, p. 535) .

Blockchain is a ledger technology, which uses cryptography, mainly public and private key encryption as well as decryption (Bahga and Madisetti 2016, p. 537) along with authoritative records of transactions (Michela 2018, p. 1). In the context of blockchain authoritative means that every user has a unique digital identity, which is imprinted on

a transaction (Polyzos and Fotiou 2017, p. 76). With blockchain every node is connected with each other. The strengths of this technology are data transparency, data immutability and autonomy (cf. Chapter 2.2.2). Data on a blockchain, once verified and accepted, is saved in a block, which is appended to the chain. Data cannot be changed after that, which is the reason why it is said, that blockchain allows for so called “trustless trust”. This can lead to problems with wrongfully added data (Michela 2018, p. 1–2) and (Bahga and Madiseti 2016, p. 535–538).

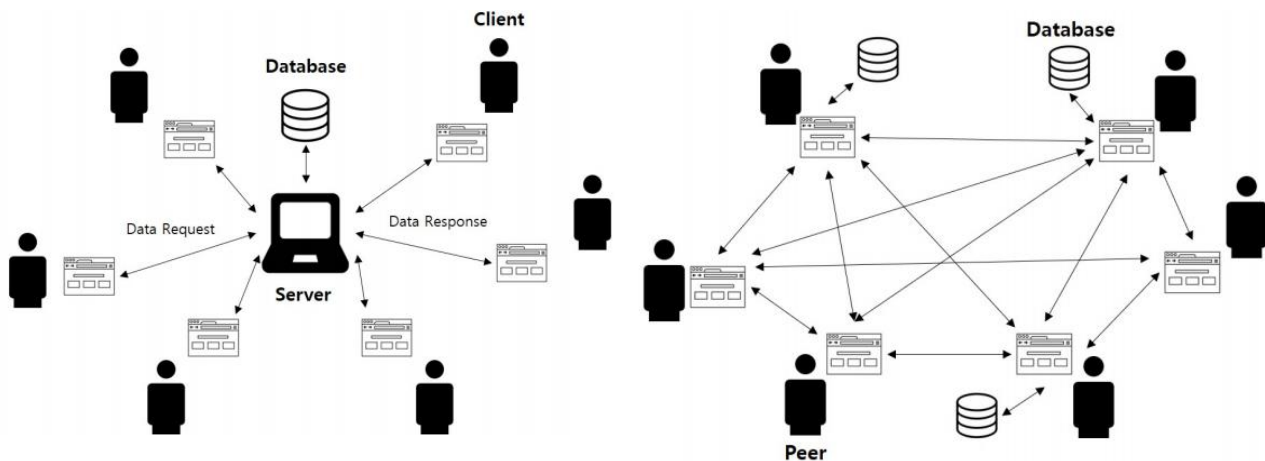


Figure 1 – Centralized server client model (left) & Decentralized peer to peer model (right) (Park et al. 2018, p. 10)

This thesis aims to analyze the technical possibilities of blockchain in smart manufacturing as well as depict the opportunities and risks of a possible blockchain adoption. In combination, blockchain can sort out some issues smart manufacturing has such as secure data collection, cyber-attacks and trust among partners.

1.2 Problem Definition and Research Questions

Blockchain technology is receiving a lot of attention in the financial industry. The reason for this is that most of its publicity comes from the cryptocurrency Bitcoin. However, blockchain has numerous other use cases. In the financial sector, examples are securities issuance, fraud prevention and notary services. There are also several non-financial use cases such as decentralized storage, where no central control unit is needed, or in the music industry to determine music royalties (Nofer et al., 2017). The manufacturing sector is another promising field for blockchain. Possible use cases for blockchain are the identification of IoT devices and using timestamps to prohibit manipulation of sensors (cf. Chapter 0).

To understand how blockchain can help in smart manufacturing and with such use cases, it is essential to explore how a blockchain works. Blockchain consists of a chain of blocks linked together sequentially (Nofer et al. 2017, p. 183–184). A block contains a list of transactions and a block header including the hash of the previous block header as well as the hash of contained transactions. Transactions are, for instance, the

transfer from one subject to another. Blockchain is known for being a decentralized database, therefore, the whole blockchain is stored on every computer. Each of the computers in the network represent a nodes (Gatteschi et al. 2018, p. 2–5). Two other essential concepts are mining and the majority consensus. The mining process starts if a new block is to be added to the blockchain. It can be seen as a contest of who can solve a complex mathematical problem, which is to find a random value that generates a result if combined with the hash of the previous blocks header and the hash of the transactions. This concept is called “proof of work”. In the next step, the result is broadcasted to the other nodes. The blockchain is extended with a new block and all local copies of the blockchain are only updated if the majority of nodes accept the solution (Lamberti et al., 2017). The advantages of such processes are the following:

- Decentralized validation
- data redundancy
- data immutability
- trust
- transparency

Data redundancy is granted, since every participant of the blockchain has a local copy of the blockchain stored. Data immutability is granted because once a block is created it cannot be altered. Trust comes from the consensus algorithm and transparency is given, since in a public blockchain everyone can read the blockchain with all its transactions. Data immutability can be a disadvantage as well, because wrong entries of a blockchain cannot be deleted. A possibility to overcome this disadvantage is the use of flags (true/false) and data sets (Lamberti et al., 2017).

One topic smart manufacturing enterprises are confronted with is managing data security (Thames and Schaefer 2017, p. 59). A compulsory function in smart manufacturing is smart maintenance to ensure productivity (Xing and Marwala 2018, p. 26). Hence, secured data sharing is a major challenge for establishing smart maintenance strategies and implementing remote-based condition monitoring systems. A secured data exchange, analytics and storage platform is required especially when multiple stakeholders are involved in the entire maintenance processes as original equipment manufacturers, machine operators (factories) and suppliers. This thesis investigates if blockchain technology may help tack this problem and explores related advantages, limitations and boundary conditions.

Summing up the problems can be categorized as follows:

1. Need for a secure data exchange platform
2. Unclear requirements concerning blockchain adaption in the sector of smart manufacturing

3. Little knowledge about risks associated to a possible blockchain adaption, which is highly relevant concerning such a long-term investment.

Therefore, the following research questions will be answered in this thesis to tackle the problems smart manufacturing enterprises have:

1. What is an appropriate way of utilizing blockchain in smart manufacturing enterprises?
2. What are the industrial requirements for employing blockchain?
3. What are the technological and non-technological risks associated with it?

An expected outcome is hard to determine, since blockchain is a relatively new technology with only a limited amount of practical usage besides in the context of cryptocurrencies. Furthermore, thinking about an appropriate way of how to use blockchain, there is not the one solution that solves all problems. However, it is expected that blockchain can close some gaps in smart manufacturing.

The main and regarding industrial requirements most important question is if and why blockchain should be used to solve a problem. If the outcome is that another technology such as a cloud solution fits better, then there is no reason to use a blockchain based approach. Hence, the starting point of why blockchain should be used will to be a critical one.

Additionally, one risk that should not be underestimated are people themselves. What do they know about blockchain and do they trust this technology? In addition, other important factors are for example, the needed infrastructure and a strategic plan for the adaption of blockchain.

1.3 Solution Approach

The problem of finding an appropriate way of utilizing blockchain in smart manufacturing enterprises can be seen as a “wicked problem” because it may only be possible to find a solution that is better, in contrast to true or false. Furthermore, the solution does not solve all problems completely. Testing the solution was difficult. Therefore, the ad hoc development approach fits best. For this purpose, the design science approach proposed by Hevner was used (cf. (Hevner et al. 2004)). The framework can be seen in Figure 2, which shows how to systematically execute and evaluate this thesis. The environment is the problem space with people, technology and organizations. Information system research is conducted by developing theories or artefacts and evaluating them. Lastly, there is the knowledge base, where foundations and methodologies can be found (Hevner et al. 2004).

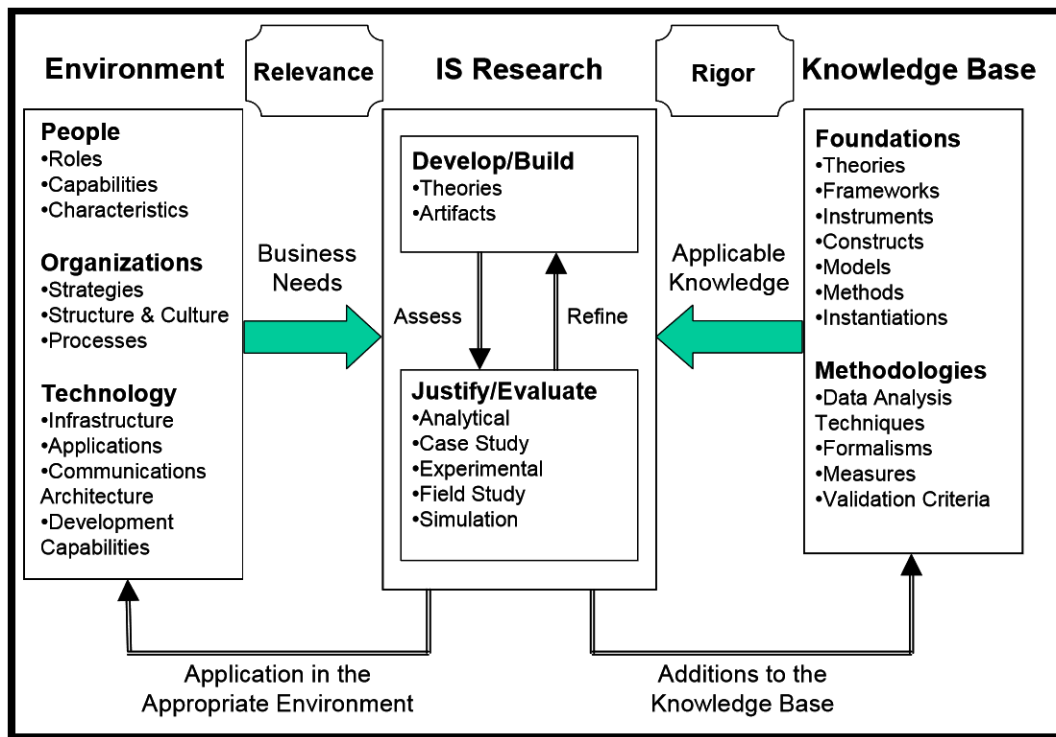


Figure 2 – Research Framework (Hevner et al. 2004)

In Table 1 the design science research guidelines for the framework can be found, which correspond to this thesis as follows:

- Guideline 1: The artefact produced will be based on the use case explained in chapter 0 and will be implemented as a proof-of-concept.
- Guideline 2: The problem at hand was explained in chapter 1.2. Since blockchain is of rising interest, it is important to analyze the possibilities of blockchain in smart manufacturing.
- Guideline 3: The evaluation will be done by comparing a state-of-the-art solution to the solution proposed in this thesis as well as by analyzing parameters for data sharing security.
- Guideline 4: The contribution of this work will be the knowledge whether or not and how blockchain should be used for data sharing in the context of smart manufacturing.
- Guideline 5: The research rigor will be kept by using the design science approach as well as getting regular feedback from my supervisor.
- Guideline 6: The best possible setup for the chosen use case will be applied, by doing literature review.
- Guideline 7: The presentation of this work will take place in the “Seminar for Master Students” as well as at the defensio.

Table 1 Guidelines (Hevner et al. 2004)

Guideline	Description
Guideline 1: Design as an Artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
Guideline 2: Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
Guideline 3: Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
Guideline 4: Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
Guideline 5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
Guideline 6: Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
Guideline 7: Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

1.4 Structure of the Thesis

The rest of this thesis is structured as follows (also see Figure 3):

Chapter 2 gives the theoretical foundation of blockchain, smart manufacturing and the combination of those topics. Challenges, advantages and disadvantages as well as use cases are outlined.

Chapter 3 provides the state of the art for this thesis which is cyber security in smart manufacturing and Industry 4.0. It shows different aspects and use cases such as collaborative work in manufacturing.

In chapter 4 the prototype, which has been developed as a proof-of-concept, is shown and evaluated.

Chapter 5 gives a discussion, outlook and possible future work on this topic including the answers to the research questions.

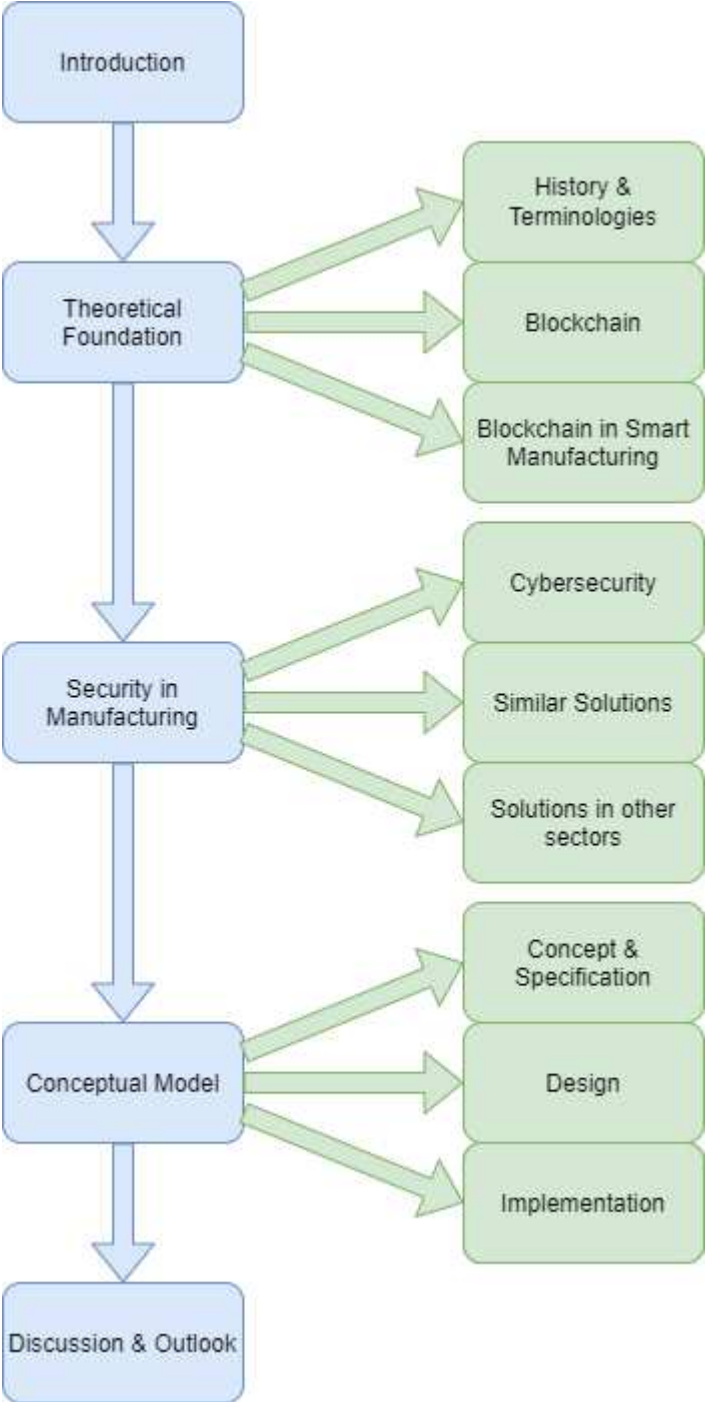


Figure 3 - Structure of the thesis

2 Theoretical Foundations

The theoretical part focuses on the topics of smart manufacturing blockchain, as well as the combination of those topics to get the background knowledge needed for the practical part. Notably, blockchain is analyzed in the context of smart manufacturing.

The first chapter discusses smart manufacturing with some definitions, a brief history, challenges, advantages and disadvantages. This is then followed by a chapter about blockchain. The last part is the combination of these two topics, which creates the foundation for the use case implementation.

2.1 Smart Manufacturing: History & Terminologies

Starting with the industrial revolution in the second half of the 18th century continued by mass production systems in the early 19th century, the origins of smart manufacturing are not too distant in the past. After the commercialization of electricity and the introduction of information and communication technology, automation systems were introduced in the late 20th century. Looking back, a fast advance in ICT as well as a broad spectrum of innovations can be observed (Kang et al. 2016, p. 111–124).

Before going into detail on the terminologies, the journey up to this point should be understood (see Figure 4). Starting with the first industrial revolution and the age of steam. At the end of the 18th century came the introduction of mechanical manufacturing systems that used the power of steam. The age of electricity or the second industrial revolution started in the late 19th century with the use of electricity for mass production. The third industrial revolution or the information age started by introducing automation and microelectronic technology into manufacturing (Xu et al. 2018, p. 2942–2943).

It seems clear that the challenges of each industrial revolution were different. In the first industrial revolution labour and the procurement of coal were a challenge, the third and fourth revolution had different obstacles. Nevertheless, there are also major differences between the third and fourth industrial revolution. ICT was only introduced in the third revolution and the focus was mainly on automation. In Industry 4.0 connectivity and decentralization reach another level with some major challenges like cybersecurity (Xu et al. 2018, p. 2943–2944).

Right now, businesses are in the phase of adapting Industry 4.0 standards and technologies. Future research on Industry 5.0 has already begun. Industry 5.0 should include people labour into the equation. A synergy between machines and the work

people do to solve not only technical, but social problems is the goal (Martynov et al. 2019, p. 539–540).

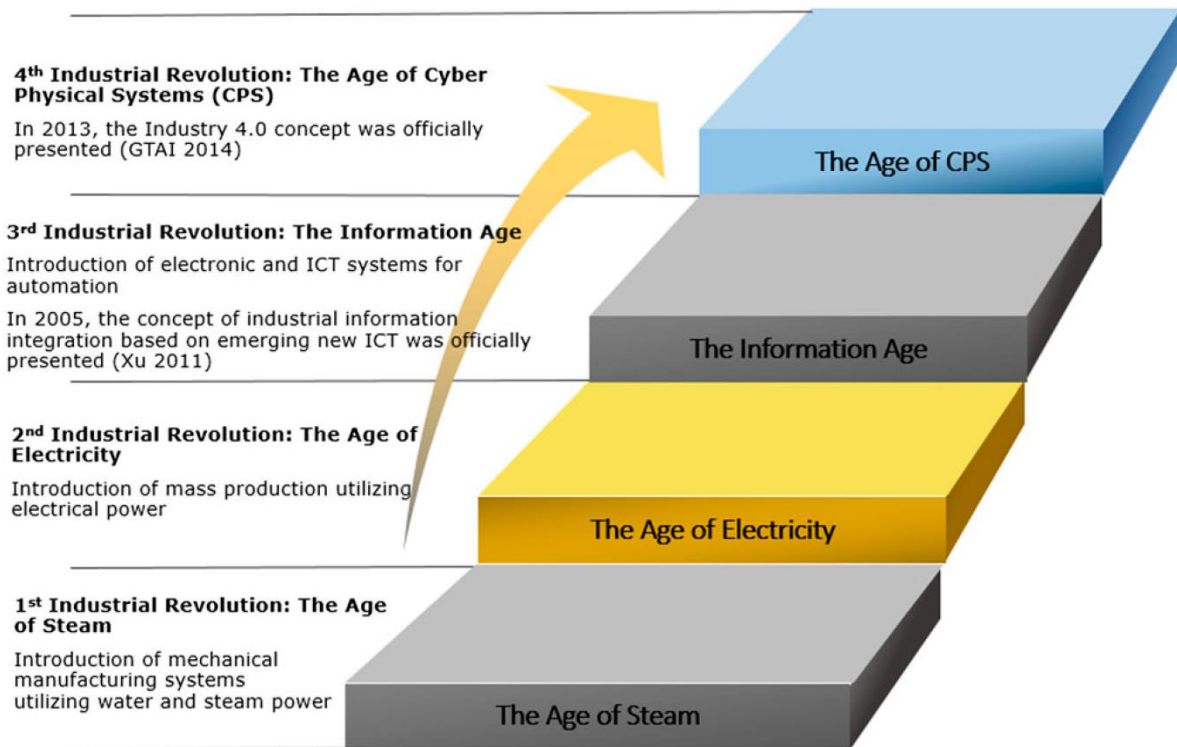


Figure 4 - Industrial revolutions (Xu et al. 2018, p. 2943)

Following, several terms will be defined and explained in order to understand the broad context of smart manufacturing.

The definition by **Industrie 4.0** Austria is as follows: “Industry 4.0 is defined as the digitalization and integration of the entire value chain and follows the mechanization, electrification and automation as the fourth industrial revolution.” (Verein Industrie 4.0 Österreich 2019). In short terms Industry 4.0 stands for the merging of production technologies with ICT technologies (Verein Industrie 4.0 Österreich, p. 11). Industry 4.0 not only can lead to cost reductions and efficiency increase, but it can also open new markets with highly innovative products (Verein Industrie 4.0 Österreich, p. 13).

Industry 4.0 can be seen as a paradigm, that turns factories into smart ones (Yao et al. 2017, p. 1). The German government said that Industry 4.0 is the ultimate realization of smart manufacturing.

Smart manufacturing is defined by the National Institute of Standards (NIST) and Technology as follows: “fully-integrated and collaborative manufacturing systems that respond in real time to meet the changing demands and conditions in the factory, supply network, and customer needs” (NIST 2017). Smart manufacturing is part of the fourth industrial revolution and represents a new paradigm with innovative ICT,

Robotics, AI, blockchain and other technologies mentioned in this thesis. For the realization of smart manufacturing, data-driven and smart technologies starting from CPS, Big Data, IoT to the simplest of components, for example, smart sensors are needed (Kang et al. 2016, p. 111–124). The objectives of smart manufacturing are to keep the manufacturers competitive, protect the environment and improve the safety of workers. These goals will be achieved through the change of product invention, design, production, logistics and sales (Li et al. 2017, p. 18).

A **smart factory** is part of the project Industry 4.0 and can be described as a production environment in which production facilities and logistics systems largely organize themselves without the need for human intervention (Kang et al. 2016, p. 115). The following technologies are needed in a smart factory (Chen et al. 2018, p. 6505–6515):

- Networked sensors
- Data interoperability
- Dynamic modelling & simulation
- Intelligent automation
- Scalable cyber security

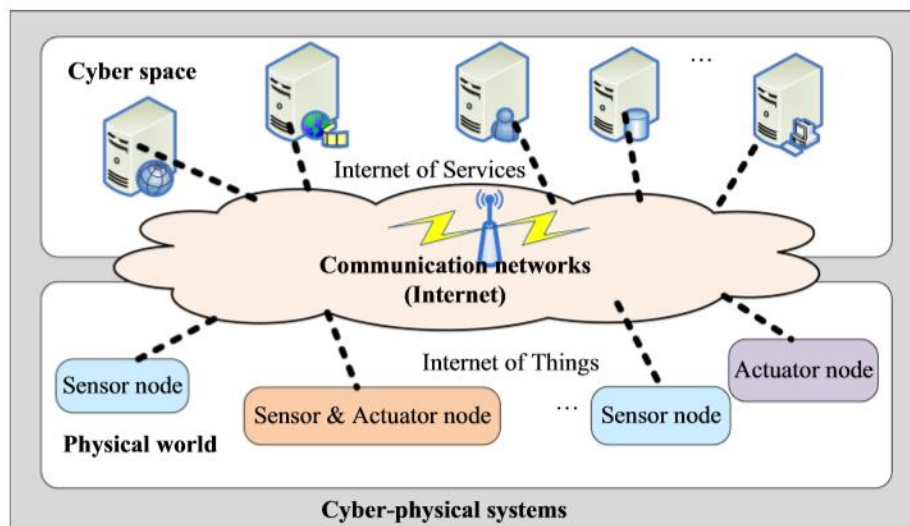
Smart sensors are one of the most critical technologies for smart manufacturing, IoT and CPS when it comes to hardware. They are on lowest level and have the purpose of collecting and controlling data in real time (Yao et al. 2017, p. 1–3) and (Lee and Seshia 2017, p. 181–200).

The term **CPS** was first used at the National Science Foundation by Helen Gill in 2006 (Lee and Seshia 2017, p. 5). CPS are complex systems built out of many heterogeneous units with a specific system structure that is used for a certain application (Kang et al. 2016, p. 118–119). There are two main functionalities. The first one is real time data acquisition, which relies on a high connectivity and information feedback. The second one is smart data management and analytic capabilities needed to create the cyberspace (Lee et al. 2015, p. 19). Figure 5 depicts an abstract example of CPS.

In CPS there are a physical world and a cyber space. In the physical world there are components such as sensors, hardware devices and actuators which are part of the Internet of Things. In the cyber space there are components like communication devices, software and all information which are part of the Internet of Services. When these two are connected, they create communication networks (Kang et al. 2016, p. 118–119).

A so-called five level architecture for realizing CPS was introduced, which can be seen as a step by step guide to developing a CPS and is shown in Figure 6 including applications and techniques associated (Lee et al. 2015, p. 19):

- The **smart connection level** is the most basic level and is responsible for data acquisition coming from sensors or ERP Systems. Important issues to consider at this level are different data types coming from different sources and the selection of the proper type of sensor (Lee et al. 2015, p. 19).
- The **data to information conversion level** is responsible for the extraction of information from the raw data collected in the previous level. The machines themselves and prognostics as well as health management are in focus (Lee et al. 2015, p. 19).
- The **cyber level** represents a central information hub because all information is delivered to it creating a network of machines. In this stage, additional analytics are needed to extract further information. Thus, machines have a way of self-comparison regarding, for instance, performance. Additionally a prediction for the future can be made by using historic data (Lee et al. 2015, p. 20).
- The **cognition level** provides the user with visualization of the information gathered in the previous two levels. The user can compare statuses of machines as well as performance data and can thereby make better decisions (Lee et al. 2015, p. 20).
- **Configuration level** acts as a resilient control system, which means it makes machines self-control and self-adaptive. This makes corrective and preventive measures possible on the basis of the cognition levels information (Lee et al. 2015, p. 20).



- **Figure 5 - View of a Cyber-physical system (Yao et al. 2017, p. 3)**

Cyber-physical production systems (CPPS) consist of cooperative and autonomous components with sub systems that are connected throughout all levels of production. There are three main factors underlining CPPS:

- Smartness

- Connectedness
- Responsiveness

Firstly, smartness is needed to be able to acquire information and also act autonomously. Secondly, connectedness lays the foundation for cooperation and collaboration not only among machines but also between machines and humans. Lastly, responsiveness is important to be able to react towards internal or external changes (Monostori et al. 2016, p. 626–628).

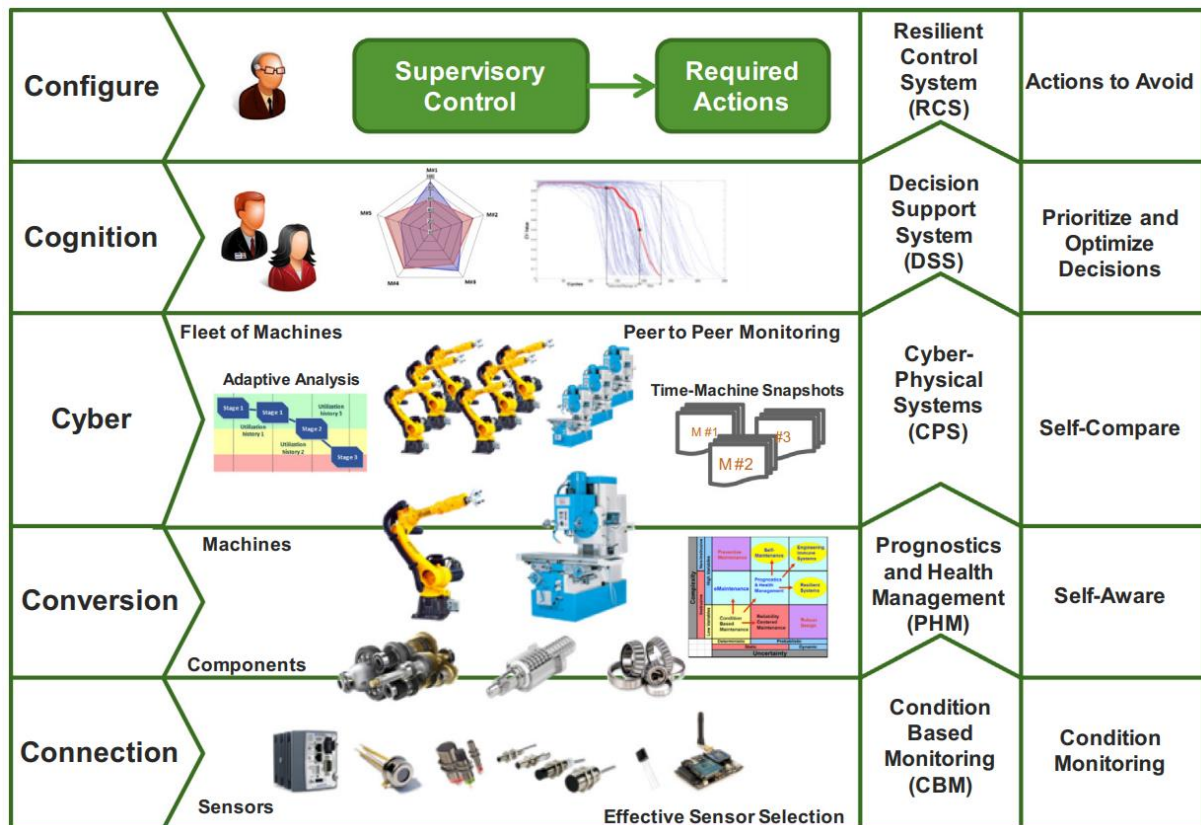


Figure 6 – Applications of the 5 level architecture for CPS (Lee et al. 2015, p. 20)

On the left side of Figure 7, the traditional automation pyramid, which was used long before Industry 4.0, can be seen. This architecture changed with modern CPPS. The control and field level still exist, since they are necessary for providing highest performance for critical control loops. The higher levels changed to a more decentralized functioning seen on the right side of Figure 7 (Monostori et al. 2016, p. 626).

Going into more detail, an eight-tuple architecture for a cyber physical production system can be described with (Yao et al. 2017, p. 2):

- Input
- Relation
- CPS

- IoS
- IoT
- Internet of Content and Knowledge
- Factory
- Output

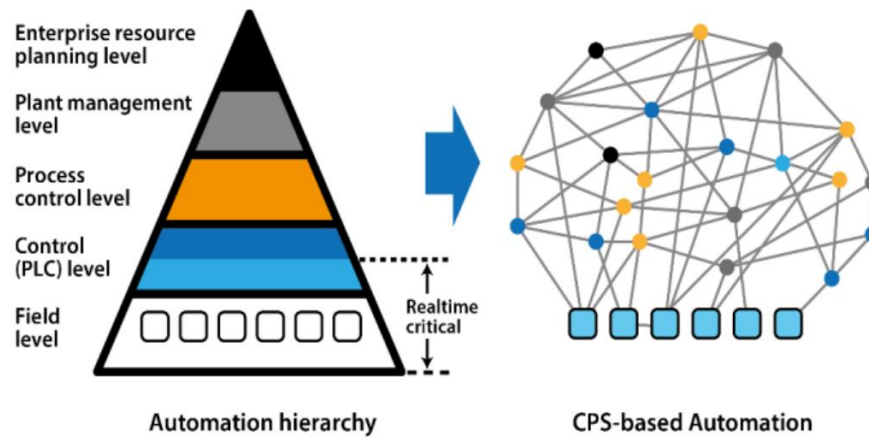


Figure 7 - Automation hierarchy and distributed services (Monostori et al. 2016, p. 626)

In Figure 8, the eight-tuple architecture of a CPS can be seen. The Input and Output represent the start and end of the process. In the center, a factory with relations to IoT and IoS can be found (Yao et al. 2017, p. 2).

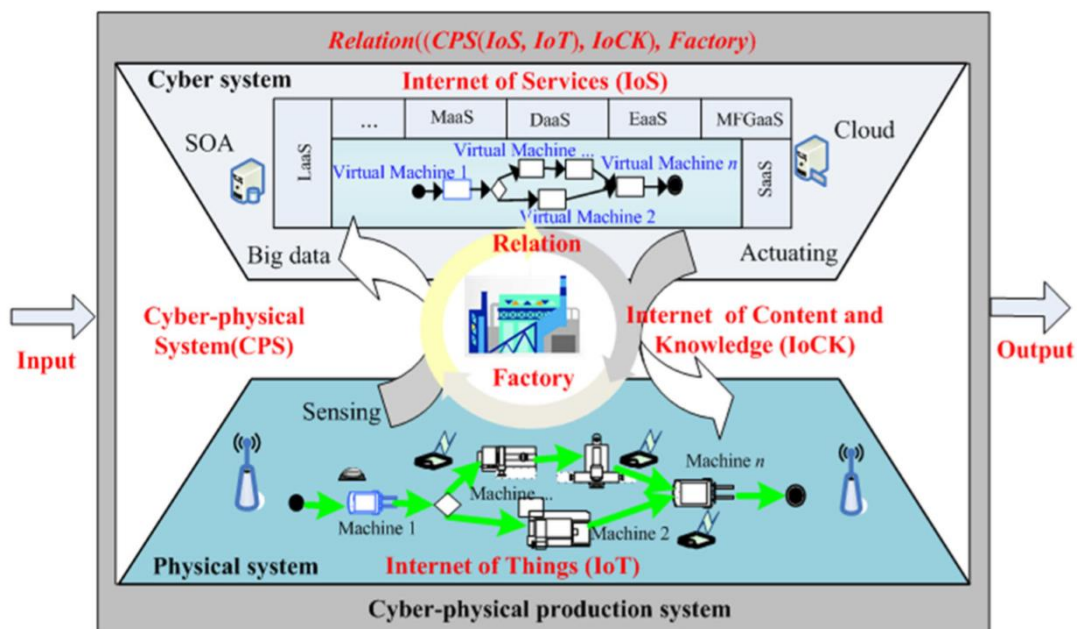


Figure 8 - CPS based manufacturing architecture – (Yao et al. 2017, p. 6)

The term **IoT** was first used in 1999 and refers to a technology that is tightly connected to CPS. In CPS the target is exchange and feedback of information for control

purposes, whereas IoT first concentrated on the identification and later also on the control of physical objects (Yao et al. 2017, p. 3). Physical objects, sensors, software and a network are part of IoT, where data is collected and exchanged. IoT can bridge between the real world and computer-based systems. The result of the usage of IoT can be increased productivity or in broader terms economy in manufacturing (Yao et al. 2017, p. 3–4) and (Panarello et al. 2018, p. 1–3).

The **Internet of Services (IoS)** can be seen as a collection of services, for example, communication, connection, interaction, collaboration and interoperation while using cloud computing visualization and web service technologies (Yao et al. 2017, p. 5).

Table 2 shows an overview of Industry 4.0 components and the respective design principle. Interoperability is a key factor for the overall success of a smart production work site. Different CPS need to be able to communicate with each other to convey information. Virtualization is realized through the creation of a virtual world, which is a copy of the physical world, by collecting data with sensors and simulating models. The capability of collecting data in real time is an important factor, to be able to react faster in case of a machine failure for instance. Modularity, or plug and play, and service orientation is realized through IoS, so that services can be accessed and used by other participants (Hermann et al., p. 11–12).

Table 2 - Design principles for Industry 4.0 (Hermann et al., p. 11)

	CPS	IoT	IoS	Smart factory
Interoperability	X	X	X	X
Virtualization	X			X
Decentralization	X			X
Real-Time Capability				X
Service Orientation			X	
Modularity			X	

In the next chapter, the future and current challenges of smart manufacturing as well as some possible solutions are presented.

2.1.1 Challenges

Several challenges arise due to the complex nature of smart manufacturing (Rajput and Singh 2019, p. 8). They can be divided into three groups: i) business model, ii) data and security and iii) operations (Rajput and Singh 2019, p. 18). Starting with challenges regarding the business model it is critical to note that the fast pace of digital disruption forces manufacturers to investigate new business models. An example is the use of mobile payment in the financial sector. Many small new businesses were

able to take market shares away from the big players caused by a slow reaction of the demand (Livesey 2016, p. 5–6).

An additional factor is the identification of profitability. Before implementing smart sensors in the whole factory, it must be analyzed where it is the most useful to collect data. Where can optimization take place and how much is the customer willing to pay for this optimization (Livesey 2016, p. 5–6)?

This is followed by a critical challenge regarding data and security, because smart manufacturing relies heavily on technology and data collection. Ensuring the secure collection and storage of data is not only a goal which manufacturers impose on themselves, but also part of data protection legislation. The more data is collected, the more data can be stolen, with the side effect of high reputational damage (Rajput and Singh 2019, p. 8–10).

Therefore, it is important to communicate with all stakeholders in the value chain to know which data is being stored, the reason why data is stored and finally getting everyone's approval (Livesey 2016, p. 6). The main challenge here is the lack of standards in place to facilitate easy ways to exchange data, since languages and data types differ. For instance, control engineers work with operational technology and focus on mission assurance, but the IT system administrators are working on the information technology side with the goal of information assurance. The problem arising is that the objectives of these two disciplines rarely align (Thames and Schaefer 2017, p. 2).

Another challenge is that a big amount of data makes businesses vulnerable to cyber-attacks. It is very difficult to guarantee that every system, every database, every communication channel and every sensor is safe and cannot be hacked.

The next challenge worth mentioning concerns operations and the need for agility. Setting up a smart factory is a long term process, hence it should be possible to exchange or update components easily (Livesey 2016, p. 6).

Connecting different systems with an end-to-end approach, which provides an overall picture, is another key challenge. Followed by the need for standardization to have the advantage of interoperability (Livesey 2016, p. 6) and (Rajput and Singh 2019, p. 8).

There are two important players who have a critical impact on standardization: The National Institute of Standards and Technology as well as the Ministry of Industry and Information technology of China. The "German Standardization Roadmap Industry 4.0" is a paper that tackles the problem of unification (Li et al. 2017, p. 18–20).

According to NIST there are four dimensions regarding the standardization architecture (Li et al. 2017, p. 18–20):

- Product
- Production
- Business
- Manufacturing pyramid

In terms of the product, standards should be implemented along the lifecycle, which starts with planning and designing, moves on to manufacturing as well as usage and ends with recycling. Examples for production standards are again located along the lifecycle, build, operation, maintenance and decommission. Businesses need standards along the supply chain. Lastly, standards on the whole manufacturing pyramid are necessary, with the enterprise level, control and data collection level, device level, cross level and manufacturing operations management level (Li et al. 2017, p. 18–20).

Finally, most of the arising challenges by smart manufacturing, are due to the fast pace of the market. With rapidly changing customer needs and the arising of new technologies, the risk of a long-term investment is difficult to calculate. This means that a quick reaction by the manufacturer is required to not miss out on opportunities (Livesey 2016, p. 6). A summary of the challenges categorized by business, data and security and operations can be found in Table 3.

Table 3 – Summary of challenges of smart manufacturing

Area	Challenges
Business	<ul style="list-style-type: none"> ➤ fast pace ➤ digital disruptions ➤ identification of profitability
Data and security	<ul style="list-style-type: none"> ➤ Data collection ➤ Technology advancements ➤ Cybersecurity ➤ Communication ➤ Standards
Operations	<ul style="list-style-type: none"> ➤ Agility ➤ Modularization ➤ End-to-End approach

2.1.2 Limitations in Manufacturing

Before going into detail on the advantages and disadvantages of smart manufacturing, the limitations of most current manufacturing will be analyzed. Investigation the speed of manufacturing, limitations can be categorized into process, system and coordination limits. Firstly materials, which are used in a certain process have limits, like for instance factors like heat. Secondly, the system has certain limitations, like the people performing actions, which underly a quality standard. Lastly, coordination is a limitation, when thinking about work spaces, which get more and more complex (Budak et al. 2011, p. 730–731). In Figure 9 the bottlenecks just discussed can be seen.

Importantly, fixing just one of these parameters, may not lead to an overall increase of production speed, since the other two factors can still be limiting.

Another significant limitation is the lack of real time data. Traditional manufacturers cannot leverage the advantages of CPS, thus, they mostly lack the ability to collect or process real time data. The consequences of that are a more difficult risk assessment and harder adaption to change and uncertainties (Yao et al. 2017, p. 3).

J.M. Allwood et al. / Journal of Materials Processing Technology 229 (2016) 729–757

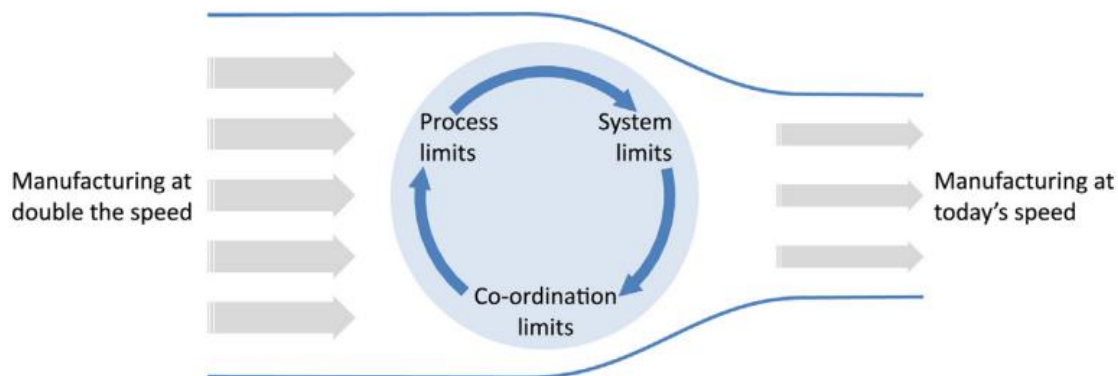


Figure 9 - Limits of manufacturers (Allwood et al. 2016, p. 730)

A further limitation traditional manufacturers face is that the components from enterprise level to production in the factory aren't interconnected. This facilitates the building of information islands on which synergies cannot be analyzed. A solution to these problems can be middleware software, but scalability is a limitation that leads to a rejection of this solution in the end (Yao et al. 2017, p. 3).

The scarcity of intelligence and proactivity also present a limitation. Control systems mostly have a fixed process with a defined cycle time. This leads to inflexibility, hence no quick reaction on a changing environment is possible (Yao et al. 2017, p. 3).

2.1.3 Advantages & Disadvantages

After discussing limitations on current manufacturing sites, the advantages of smart manufacturing will be evaluated.

One of the biggest advantages of smart factories is the access to real time data. To overcome the problem of traditional manufacturers, where planning takes place on a macro level, CPS uses pervasive sensing and connects the virtual world to the physical (Xu et al. 2018, p. 2947). The ability of real time data acquisition leads to better decision making because more information is being collected. Diagnostics, performance indicators and traceability are greatly enhanced, which leads to an overall higher agility and robustness (Yao et al. 2017, p. 3–4).

A further advantage of smart manufacturing is decentralized decision-making. Since the industrial environment is becoming more and more decentralized, a new service orientated generation is needed (Hermann et al., p. 11–12). This paradigm enables interaction across systems and devices not caring about location. This allows manufacturers to collaborate interoperably between distributed components. The result is having better control of production processes (Yao et al. 2017, p. 3–4) and (Xu et al. 2018, p. 2943).

Additional advantages are reconfigurability and interoperability which have been mentioned above. To actualize these advantages smart manufacturing uses cloud computing and web services which can be used decentralized are scalable and modular. Reconfigurability also comes from the fact that subcomponents can be built, which are easily upgraded or exchanged (Hermann et al., p. 11–12).

Proactivity and intelligence are major advantages of smart manufacturing. A vast number of components which are autonomous, fault-tolerant and reusable have the ability to initiate collaboration and interact to reach a common goal. Machines could predict failures before they occur and react to them proactively. The extraordinary trait of smart manufacturing is that it is possible to have autonomous units, that control maintenance or repairs. In smart factories units can follow the principle “predict and prevent” instead of “fail and fix” (Yao et al. 2017, p. 3–4).

To conclude this chapter, the disadvantages of smart manufacturing are discussed. One disadvantage is the adaption cost. For small or middle-sized companies, it takes a lot of investment to grow into a smart factory. In short and midterm scenarios, the costs will heavily outweigh the benefits. In a long-term scenario, the advantages and synergy effects of smart manufacturing will prevail.

Another disadvantage or challenge is the complexity of the systems. The question arises if there will be a person who can fully comprehend the decision making of all the units (Manufacturing Lounge 2018). A summary of all advantages and disadvantages can be found in Table 4.

Table 4 - Advantages and disadvantages of smart manufacturing

Advantage	Disadvantage
<ul style="list-style-type: none"> ➤ Real time data ➤ Decentralized decision-making ➤ Reconfigurability ➤ Interoperability ➤ Proactivity 	<ul style="list-style-type: none"> ➤ Adaption cost ➤ High investment ➤ Risk ➤ Complexity of systems

2.2 Blockchain

2.2.1 History & Applications

Blockchain is attracting a lot of attention since the rise of cryptocurrencies starting with Bitcoin. Therefore, the financial industry started experimenting with blockchain and possible use cases. However, there are many other possible application areas as will be shown in this thesis (Nofer et al. 2017, p. 183–187). According to a study by Deloitte, 53 percent of senior executives see blockchain technology among their top five strategic priorities. 86 percent consider blockchain as a scalable technology that will reach mainstream adoption (Budman et al. 2019, p. 3–4).

The origin of blockchain can be dated on the same day Bitcoin was created. On August 18th 2008 the domain bitcoin.org was registered. However, the idea of cryptocurrencies started much earlier namely in the year 1998, where it was found on a mailing list by Wei Dai. The first proof of concept for Bitcoin came from Satoshi Nakamoto (<https://bitcoin.org/de/faq>, 2019). After the creation of Bitcoin, three evolutionary steps of blockchain can be observed (Lamberti et al. 2017, p. 1–3).

Blockchain 1.0 started the evolutionary process with heavy ties to cryptocurrencies. The applications during this time mostly used blockchain to record transactions. Users have a digital wallet to store their credentials and transfer their money. Bitcoin was only the first cryptocurrency, followed by many more: Ethereum, Monero and Ripple to name a few well-known ones (Lamberti et al. 2017, p. 1–3).

Blockchain 2.0 represents a further step of evolution. The most important change is the introduction of smart contracts. These are pieces of code stored on the blockchain which can trigger a certain behavior if defined conditions are met. The horizon changed from simple cryptocurrencies to the managing of contracts and properties, which led to more sophisticated applications.

Blockchain 3.0 widened the horizon even more including sectors like health, science, government and others. Through all the advantages described in chapter 2.3.2, problems with censorship, anonymous voting, recording of genomic data and many others can be solved (Lamberti et al. 2017, p. 1–3). Blockchain 3.0 is the first step in the evolution that can be used by the broader society, because of its new applications. This can be seen in Figure 10.

Increasing technical complexity and social difficulties come hand in hand with the evolution of blockchain. Starting with a low complexity with Blockchain 1.0 and rising through time and iteration. Blockchain 2.0 and 3.0 promise the biggest business impact: changing business paradigms as well as processes (Hughes, Alex et al. 2019, p. 6–7).

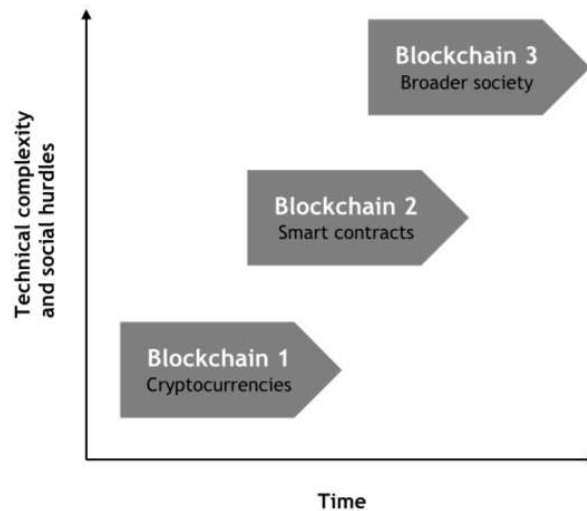


Figure 10 - Blockchain adaption (Hughes, Alex et al. 2019, p. 6)

Looking into the future Accenture predicted that between the years of 2018 and 2024 there will be a phase of growth, during which general conditions will be defined, as well as network effects and first market share shifts will appear (Höltmann and Ogyan 2019). IBM supports this statement with a forecast from a market share of \$708 million in 2017 rising to \$60.7 billion in 2024 (WinterGreen Research 2018, p. 20). According to Accenture in the year 2025 blockchain will reach the stage of maturity and play an integral part for businesses (Höltmann and Ogyan 2019).

Due to a variety of advantages shown in chapter 2.1.3 blockchain appeals to various sectors. A small list of already existing prototypes in certain sectors and contexts of blockchain can be found in Table 5.

After a brief overview of the history of blockchain, the next chapter will describe the challenges blockchain technology brings. Since we are only in the first iteration of blockchain with cryptocurrencies, the biggest challenges are yet to come with the second and third iteration.

Table 5 - Existing blockchain prototypes (Lamberti et al. 2017, p. 5–6) and (Gatteschi et al. 2018, p. 6)

Sector/Context	Description
Internet of Things	In the field of IoT blockchain is used to support higher interoperability between devices and insuring trust between the involved parties.
Healthcare	In the context of healthcare, blockchain can be used to save vital data and location information. In case of an emergency an alert can be sent.

Sector/Context	Description
Government	In the government sector, citizen votes could be collected in a verifiable but also privately manner. In that case a vote is represented by a transaction.
Software	Blockchain could make it impossible for hackers to delete or change data like logs.
Finance - Trading	Blockchain can be used to make sure a lottery winner gets his money automatically using smart contracts.
Personal data management	The identity of a user could be saved using a blockchain wallet. Instead of having to use traditional credential. The user can login with a unique identifier which can be used on other platforms as well.
Intellectual property	Blockchain solutions could license an author's work and the author can also receive payment automatically if someone else uses it. This could be done using smart contracts.
Internet	Blockchain can be used to reduce censorship, by using advantages like immutability, which makes it impossible to change data once published.

2.2.2 Definition

Blockchain is a decentralized distributed ledger that uses network nodes and records transactions which are enforced between these nodes. The main features of blockchain technology are immutability, cytological identity, integrity, durability and pervasiveness (Nofer et al. 2017, p. 183–184). In Figure 11 the basic functionality of a blockchain is represented.

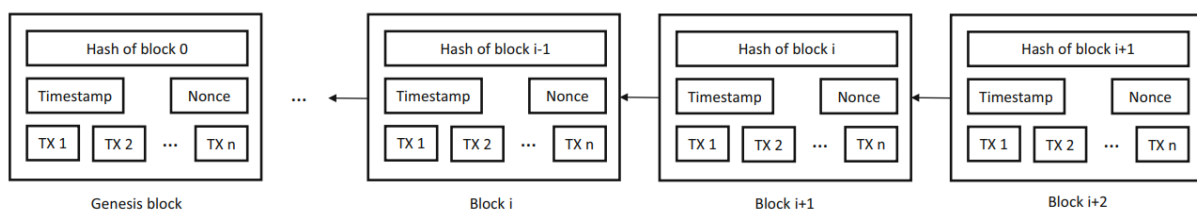


Figure 11 – Blockchain example (Nofer et al. 2017, p. 184)

A so-called genesis block which represents the first block of the chain and three other blocks can be observed as part of a blockchain. In each block there are several transactions stored (TX 1, TX 2, ..., see Figure 11). A block can be added to the chain only if the majority of nodes in the blockchain network agree. A block always points to its predecessor. The result of this architecture is that a complete history of all transactions is saved and can be accessed. Furthermore, a block contains a

timestamp, a nonce as well as a unique hash value of the previous block (Nofer et al. 2017, p. 183–184).

To narrow down the terms used above, Table 6 represents definitions of important terms used in connection with blockchain.

Table 6 - Blockchain definitions

Terms	Description
Node	A node represents a copy of a blockchain that is stored on a computer in some network(Gatteschi et al. 2018, p. 2–5).
Transaction	In terms of cryptocurrencies, a transaction can be a transfer of the currency from one user to another one. Another example could be a chat between two users(Gatteschi et al. 2018, p. 2–5).
Mining	The mining process has two parts. The first one is to check previous transactions and the other one is to create new blocks by solving a predefined complex mathematical problem(Gatteschi et al. 2018, p. 2–5).
Nonce	In terms of blockchain, a nonce is used to set the level of difficulty in the process of solving the mathematical problem(Gatteschi et al. 2018, p. 2–5).
Block	A block groups and stores transactions in a certain timeframe, saves the hash value of the previous block and the nonce (Gatteschi et al. 2018, p. 2–5).
Majority-consensus	The majority consensus sets the rules to how decisions are made. Mostly used is the so-called Proof Of Work algorithm, as described in the mining process (Gatteschi et al. 2018, p. 2–5).
Smart contract	Peace of code located on a blockchain that is identified by a unique address. It consists of executable functions and variables (Bahga and Madisetti 2016, p. 536) .
Ethereum	Is a type blockchain platform defined in Table 8.
Ether	Ether is the currency of the Ethereum blockchain and is the reward for mining in the Ethereum network. Wei is the base unit for Ether where 1 Ether = 10^{18} Wei (Bahga and Madisetti 2016, p. 538).
Public & private Key	Each account has a public and a private key, which is stored in a JSON file. The public key is derived from the accounts address and the private key is encrypted with the accounts password (Bahga and Madisetti 2016, p. 537).
Gas	Gas can be seen as the fuel for an Ethereum blockchain. Executing operations on a blockchain costs gas. The sender of a transaction defines the amount of gas price. If this price is high enough, the

Terms	Description
	transaction is executed, and the balance is refunded. The gas fee corresponds to the work needed to execute a transaction regarding the number of atomic instructions (Bahga and Madiseti 2016, p. 538).

2.2.3 Consensus algorithms

A consensus algorithm is a mechanism by which a blockchain network reaches consensus. This means that other participants need to agree before a blockchain participant can add a block to the chain or make any other updates (Vukolić 2016, p. 112–114). Table 7 shows, an overview of different consensus algorithms with their characteristics. For instance, “Proof of Work” runs on a permission-less blockchain type with probabilistic transactions, a low transaction rate, a high scalability of network, an adversary tolerance bigger than 25% of the computing power, an open node identity management and a high energy consumption. Depending on the requirements and environment, one should choose the best fitting consensus algorithm (Lang et al. 2018, p. 1096).

Table 7 - Consensus algorithms (Lang et al. 2018, p. 1096)

Consensus Algorithm	Characteristics					
	Blockchain Type	Transaction Rate	Scalability of Network	Adversary Tolerance	Energy Requirements	Typical examples
Proof of Work (PoW)	Permission-less	Low	High	< 25% of computing power	High	Bitcoin / Litecoin / Ethereum (until 2018)
Proof of Stake (PoS)	Both	High	High	< 51% of stake	Medium	Tendermint / Ethereum (from 2018)
Delegated Proof of Stake (DPoS)	Both	High	Medium	< 51% of validators	Low	BitShares / EOS / Lisk / Ark / Steem
Byzantine Fault Tolerance (BFT)	Both	High	Medium	< 33,3% of fault replicas	Low	Practical BFT / Federated BFT
Proof of Elapsed Time (PoET)	Both	Medium	High	< 25% of computing power	Low	SawtoothLake / HyperLedger Fabric
Proof of Activity (PoA)	Permissioned	Medium	High	< 25% of computing power	High	Decred / SIKKA
Proof of Burn (PoB)	Permission-less	Medium	Medium	< 51% of stake	High	Slimcoin / XCP
Proof of Capacity (PoC)	Permission-less	Medium	Low	< 25% of computing power	Medium	Burstcoin / SpaceMint

After these definitions, it is important to mention the types of blockchains, namely public, private or consortium. Bitcoin is the most prominent example of a public blockchain, where the blockchain can be read and extended by everyone. In a private blockchain, only the organization members have writing permissions and reading permissions. These rights can or cannot be given to other members. The last type is the consortium blockchain, where a set of defined nodes control the validation process and share information (Gatteschi et al. 2018, p. 2–5).

Depending on the use case at hand, one of the three types fit best. On the one hand public blockchains offer the advantages of full decentralization. On the other hand, private and consortium blockchains have lower validation costs, lower time needed for the validation process, enhanced privacy as well as a reduced risk of attacks. However, it needs to be mentioned that a hybrid of those types could be interesting for certain applications as well. An example could be a private blockchain for the back-end and a public blockchain for payment (Gatteschi et al. 2018, p. 2–5).

In the next chapter, different blockchain implementations are discussed and described to get a better understanding on which technologies are on the market.

2.2.4 Blockchain implementations

In early 2020 there are several implementations or platforms for blockchain. In this chapter, three different implementations are presented in Table 8.

Table 8 - Blockchain implementations

Blockchains	Description
Ethereum	Ethereum is a decentralized open source blockchain platform, which can be used for any kind of an agreement or transaction. Ethereum is as of its basis a permissionless public blockchain but can be implemented privately as well. Smart contracts can be compared to autonomous agents in Ethereum. They can read and write data, perform computations as well as call other contracts. Ethereum uses the PoW consensus algorithm, in which miners have to solve a cryptographic puzzle to verify transactions and create new blocks ¹² .
Hyperledger	Hyperledger is an open source permissioned blockchain infrastructure hosted by the Linux foundation. It can be used in a variety of fields, for instance finance, healthcare and supply chain. There are different platforms such as the Hyperledger Fabric or Hyperledger Sawtooth. Hyperledger Fabric enables the use of smart contracts as chain codes. Chaincodes can be written in the languages Go or JavaScript. They are executed by validation nodes inside a docker ³ .
Quorum	Quorum is an Ethereum based permissioned blockchain protocol which was designed for the financial sector to support privacy. Quorum also uses a fork of the geth client. The major features are contract privacy, permission management and a multiple voting-based consensus algorithm ⁴ .

¹ <https://www.ethereum.org/developers/>

² <https://solidity.readthedocs.io/en/v0.5.11/>

³ <https://www.hyperledger.org/about>

⁴ <https://github.com/jpmorganchase/quorum/wiki>

2.2.5 Challenges

Challenges can be categorized in both technical and social nature. The biggest social challenge in early 2020 is that most people do not understand what a blockchain is and what it can be used for. The consequence is a lack of trust, which reflects in the people's willingness to use blockchain technologies. Making the concept of blockchain as well as the usage of blockchain technology more understandable could help to overcome this challenge (Hughes, Alex et al. 2019, p. 7–8).

Technical issues also play a crucial role, since it needs a considerable amount of research and development by start-ups, software engineers and venture capitalists to solve still existing questions regarding scalability for instance. It is essential to find solutions to those questions to support maturity and to broaden the application contexts (Hughes, Alex et al. 2019, p. 7–8).

The variability of the transaction speed of a blockchain is something to keep in mind. Depending on the use case at hand, one has to choose the right type of blockchain or at least keep the factor in mind. In the case of blockchain about seven transactions per second can be handled. Another example would be the blockchain EOS where up to 3.000 transactions per second can be computed (Hughes, Alex et al. 2019, p. 7–8).

Governance is another big challenge to overcome for blockchain adaption. To profit from an advantage such as interoperability, standards and agreements on a global level are needed. This creates the next challenge which is the lack of legislation. It takes a vast amount of time to make changes in legislation, while technological innovations happen faster. There are a lot of open issues with topics such as ownership, lawfulness of smart contracts (Gökalp et al. 2018, p. 181–182).

Transparency and privacy have to be kept in an equilibrium. For instance, in the healthcare sector, privacy plays a big role and transparency isn't always desired (Gökalp et al. 2018, p. 181–182).

Sustainability and scalability are also some challenges to keep in mind. If the private key of one's wallet is lost, there is no way of retrieving it again. Due to the growth of the blockchain, more and more computational power as well as data storage has to be used (Gökalp et al. 2018, p. 181–182).

To find out whether or not it is worth to face these challenges, one should ask the following questions:

- Is a shared database needed?
- Should multiple parties use saved data?
- Do I not trust one of these parties?
- Is it important to see all the transactions and how they are linked to each other?

If some of these questions are answered with no, then the usage of blockchain should be reevaluated (Gatteschi et al. 2018, p. 7–8). In Table 9 a summary of challenges of blockchain adoption can be found.

Table 9 - Blockchain challenges

Type of Risk	Risk
Social	Misconception
	Missing trust
	Willingness
Technical	Scalability
	Transaction speed
	Governance
	Legislation
	Transparency versus Privacy
	Sustainability

In the following two chapters the limitations, advantages and disadvantages of adapting blockchain will be discussed.

2.2.6 Limitations

Blockchain technology has some prominent limitations compared to traditional solutions (Hughes, Laurie et al. 2019, p. 119). However, it has to be said that the limitations highly depend on the context where and how a blockchain is used as well as the type of blockchain used (Mendling et al. 2018, p. 4–5). Nevertheless, the following table summarizes some limitations of blockchain that can be applied universally.

Table 10 - Limitations of blockchain technology

Limitation	Description
Security model	Although the use of a public and private key is a highly secure process, the loss or the theft of one's private key can have serious consequences (Hughes, Laurie et al. 2019, p. 121).
Usability	Usability is a limitation most concerning first time users. Due to the complexity of blockchain networks and smart contracts

Limitation	Description
	it is not easy to start using this technology (Mendling et al. 2018, p. 5).
Lack of privacy	Each participant of the blockchain has access to the network's transaction data. This leaves a lack of privacy, but there is a development such as channeling to change that (Hughes, Laurie et al. 2019, p. 121).
Standardization	The lack of standardization is also limiting blockchains potential. Examples are the standardization of metrics for scalability, performance and security. Comparing different blockchain solutions will be impossible (Udokwu et al., p. 13).
High costs	The advantages of blockchain mechanisms such as the PoW consensus algorithm seem clear, but they also can be a limitation. The cost of having a copy of the whole blockchain on each node in addition to the cost of mining is a limitation (Udokwu et al., p. 15).
Governance	The distributed nature of blockchain can again be of great benefit, however in some use cases a certain amount of control is needed (Hughes, Alex et al. 2019, p. 121).

2.2.7 Advantages & Disadvantages

A number of the advantages of blockchain originate from the usage of this technology in the context of cryptocurrency. Security mechanisms like asymmetric cryptography play an important role in that regard (Laabs and Đukanović 2018, p. 145–146). These and other advantages of the adoption of blockchain are listed in Table 11.

Table 11 - Advantages of blockchain

Advantages	Description
Trust	Trust or rather a lack of trust is one of the most influential parts of blockchain. There are a few aspects of the blockchain ecosystem that ensures trust between parties. Influential in terms of money transaction is the removal of the need for intermediaries such as banks. Usually, a bank is required as a trusted partner to ensure that no one is being betrayed if money is being transferred from account A to account B. This part isn't necessary anymore when using blockchain, due to other advantages such as immutability and transparency, which will be explained later (Hughes, Laurie et al. 2019, p. 120–121) and (Laabs and Đukanović 2018, p. 146). The second trust ensuring mechanism is the use of a consensus algorithm in the combination with the usage of

Advantages	Description
	cryptography. These measures make transaction and interactions more save as well assure that one's own identity is not stolen or depraved (Gatteschi et al. 2018, p. 4–5).
Decentralized validation	Decentralization brings advantages such as data redundancy and decentralized validation leads to the fact that no intermediaries are needed. This again brings us back to the big advantage of trust (Hughes, Alex et al. 2019, p. 4).
Data-immutability	Data immutability means that a transaction written in a block cannot be changed. This can be seen as an advantage, in the case of someone trying to hack a system but also as a disadvantage when wrong data is added to the blockchain. It has to be said that the advantage is pretty big, since one way of attacking a system is to change data, which isn't possible with blockchain technology (Laabs and Đukanović 2018, p. 146).
Data-redundancy	Due to the decentralized way of how blockchain works, every node has a copy of the whole blockchain saved. This means that there are a lot of copies in the network, which leads to a high data redundancy, which prevents data loss (Gatteschi et al. 2018, p. 4–5).
Transparency	Everyone can read data stored on the blockchain, as well as every transaction on it in case of a public blockchain. In case of a private blockchain, only nodes with the right credentials have access to the data. This still leads to a high transparency, since they again can have access to all data (Gatteschi et al. 2018, p. 4–5) and (Laabs and Đukanović 2018, p. 146)
Secure Transaction	Blockchain encrypts and decrypts data using techniques such as public key cryptography (Hughes, Alex et al. 2019, p. 4).
Interoperability	A common data format is used with different data sources (Gökalp et al. 2018, p. 179–181)
Higher automation	A higher level of automation can be achieved with the help of smart contracts. Certain processes could be started automatically with the help of conditions (Lamberti et al. 2017, p. 8–9).

After having discussed the advantages of blockchain, the disadvantages of a high level

of security will be explained. Also, other issues a possible blockchain adaption can bring are listed in Table 12.

Table 12 - Disadvantages of blockchain

Disadvantages	Description
High power consumption	The bigger the network of nodes is, the higher the power consumption is. When using Bitcoin, one transaction is to be presumed to cost about \$6 (Lamberti et al. 2017, p. 8–9).
Mining	The problem with mining is that it requires expensive hardware and most of the computing power is wasted, since finding a new block is a competition. This disadvantage could be circumvented if, instead of a proof-of-work concept, a proof-of-stake concept is used (Lamberti et al. 2017, p. 8–9).
Data-replication	Can also be seen as a disadvantage, since it requires more space to store all the data in the network. In case of Bitcoin, it requires 170 GB of storage on each node (Lamberti et al. 2017, p. 8–9).
Scalability	Considering the transactions that can be handled per second, blockchain solutions do not compare too well with traditional systems. The main reason for that is the need for computing power to solve the complex problem. It has to be said that other concepts are in development such as reducing the complexity of the problem and only letting “trusted” nodes mine (Gatteschi et al. 2018, p. 9–10) and (Dieterich et al., p. 11–12).
Usability	Usability is also a disadvantage and limitation for blockchains right now. Besides the already mentioned social challenges with blockchain, aspects such as the loss of your credentials are major issues. In case of cryptocurrencies, one could lose the access to ones funds (Gatteschi et al. 2018, p. 9–10).

Lastly, disadvantages such as the speed of adding information, which can be slow in a network such as bitcoin, and also the issue unchangeability needs to be mentioned. Immutability can be a big advantage when talking about fraud prevention, but what about wrongly added data (Lamberti et al. 2017, p. 8–9)?

2.3 Blockchain in Smart Manufacturing

After analyzing blockchain technology and smart manufacturing separately, the following chapter will combine these two topics, give an introduction and discuss the

challenges, advantages as well as disadvantages. In the last subchapter, potential use cases will be investigated.

Distributed security is a fundamental feature of blockchain that can solve one of the key challenges in smart manufacturing, i.e. a secure way of record transactions between distributed components (Mohamed and Al-Jaroodi 2019, p. 854–855). Blockchain provides several features to solve challenges such as cryptography, digital identities and replication. By the very nature of blockchain technology, where every new verified block is linked with the one created before and every transaction is saved in these blocks, it is impossible to alter one of these blocks. Due to unique identities it can always be ensured, that the users involved in an agreement can be traced back. The result is a better protection of transactions with a lower risk of exposure to hackers and a higher level of trust between users of blockchain solutions (Mohamed and Al-Jaroodi 2019, p. 854–855) and (Shrier et al. 2016, p. 6–8).

The next feature mentioned are digital identities. A digital identity can be seen as a passport which blockchain uses to identify a person or an organization. A digital identity can also be expanded to save properties or other objects (Underwood 2016) and (Shrier et al. 2016, p. 5–7). However other solutions for creating digital identities already exist, for instance, in Austria, you can setup a digital identity to sign documents (Mohamed and Al-Jaroodi 2019, p. 854–855).

Smart contracts are also highly relevant in the context of smart manufacturing. They enable contracting over a public network without the need for a third party, since blockchain technology provides trust between the parties. A smart contract is secure, can be tracked and cannot be changed. For industrial use, smart contracts can advantageous in terms of automating agreement processes between firms and their suppliers, which results in reduced administrative costs. They represent a more efficient way of managing contracts (Luu et al. 2016, p. 254–257). There are many possible scenarios in which such smart contracts can be useful for smart manufacturing. Examples for this are contracts for transportation, suppliers, distributors, subcontractors and many others (Mohamed and Al-Jaroodi 2019, p. 854–855).

Finally, the last important feature blockchain technology can provide for smart manufacturing are micro controls. Blockchain enables the ability to do micro metering, micro measurement and with that also micro adjustments in a dynamic manner. Since it is possible to securely store data and transactions, the amount of stored data will rise and the knowledge about the company and its processes as well. Due to the analysis of this data, better quality controls can be implemented. The immutability of data on the blockchain is an advantageous feature for audits and evaluations. In smart manufacturing, data can be recorded, analyzed and controlled over the whole value chain (Mohamed and Al-Jaroodi 2019, p. 854–855).

Looking at the challenges of smart manufacturing and the benefits of blockchain technology, several overlaps can be found. A mapping of the challenges mentioned in chapter 2.1.1 and the benefits of blockchain mentioned in chapter 2.2.7 can be found in Table 13.

Table 13 - Mapping challenges of smart manufacturing to blockchain benefits

	Trust	Decentralized validation	Data immutability	Data redundancy	Transparency	Secure transaction	Interoperability	Automation
Secure data collection		x				x		
Resistance against cyber attacks		x	x	x	x	x		
Depict whole value chain	x				x	x	x	
Agility					x		x	
Trust among partners	x	x	x		x	x		
Profitability							x	x

Blockchain enables secure data collection by the decentralized validation mechanism as well as the cryptologic means of blockchain. In addition, data immutability, data redundancy and transparency are useful tools against cyber-attacks. Trust and transparency are the main features which enable an enterprise to depict the whole value chain in order to be able to make better decisions. Agility is granted through transparency and interoperability. However, profitability is a challenge. Smart contracts can tackle this challenge through a higher level of automation. Almost all of the benefits mentioned account for trust among business partners (Laabs and Đukanović 2018, p. 146–149)

When it comes to the adoption of blockchain some essential decisions that have to be made. What blockchain should be used? What type? What consensus algorithm is best? To answer these questions, one must look at the use case at hand and analyze the ups and downs of each approach.

Considering all the challenges and advantages as well as disadvantages of smart manufacturing, the effects of a possible blockchain adoption will be analyzed in the following chapters. Can blockchain technology benefit smart manufacturing?

2.3.1 Challenges

This chapter will investigate challenges the adaption of blockchain technology brings in the context of smart manufacturing. In Table 14 a SWOT analysis can be found in order to get a better overview of the strengths as well as weaknesses, opportunities and threats of blockchain adaption.

Table 14 - SWOT analysis: adopting blockchain (Gatteschi et al. 2018, p. 9–11)

		Internal Analysis	
		Strengths	Weaknesses
External Analysis	Opportunities	<ul style="list-style-type: none"> - No intermediaries needed - Automation trough smart contracts - Transparency - No data loss or falsification - Accessibility - Security 	<ul style="list-style-type: none"> - Scalability - Energy consumption - User privacy - Loss of credentials - Autonomous code → attackers - -Regulation
	Threats	<ul style="list-style-type: none"> - Addressing new markets - Competitive advantage - Large amounts of data coming from different kinds of actors in the supply chain 	<ul style="list-style-type: none"> - Long term investment - Not applicable for all processes - Government could consider blockchain dangerous - Not fully researched

The strengths blockchain technology provides are mainly technical related. The main strengths relevant to smart manufacturing are the secure way of collecting and storing data due to blockchains cryptographical mechanisms. The storing on every node in the network brings redundancy, which prevents data loss, as well as transparency, because every node has access to all data. The immutability of data on the blockchain leads to security, since data cannot be falsified (Fernández-Caramés and Fraga-Lamas 2019, p. 13).

In terms of weaknesses, the most relevant are scalability, energy consumption and performance. All those weaknesses depend on the type of blockchain and the consensus algorithm used. A smaller private blockchain has a lot less energy consumption than Bitcoin, which needs about 170 GB for storage and cost about 7 \$ per transaction (Gatteschi et al. 2018, p. 9–11). The main factors that influence the

energy efficiency are whether a mining process is carried out or not, inefficient P2P protocols and the complexity of the cryptographic algorithms. These factors can be handled by large servers, but for small devices such as battery powered devices this is definitely a challenge (Fernández-Caramés and Fraga-Lamas 2019, p. 13).

Scalability can also differ depending on the data saved on the blockchain, but due to redundancy this is an important issue to keep in mind. The number of transactions which can be handled by a blockchain is much lower than traditional systems can. This comes from the fact that every block of the chain must be validated. Nevertheless, if compared to how long a money transfer at a bank takes, which is usually one or two days, blockchain is still much faster in terms of its cryptocurrency roots (Gatteschi et al. 2018, p. 9–11) and (Dieterich et al., p. 11–12).

However, the dark side of transparency can do harm to the user's privacy. By analyzing the transactions of a user, one could look at the incoming transactions and thereby find out the amount of money owned for instance. Solutions to the anonymizations of transactions are currently being researched (Gatteschi et al. 2018, p. 9–11) and (Laabs and Đukanović 2018, p. 146).

The concept of smart contracts with a self-executing code can also be seen as a threat, because hackers could benefit from it. An example is an attack, where hackers stole \$60 million from the Ethereum network in 2016. Even if the written smart contract is bug free, it is sometimes necessary to inject information from outside, which could lead to potentially harmful data entering the blockchain (Gatteschi et al. 2018, p. 9–11).

Additionally, blockchain technology is not fully mature so far, which makes it vulnerable to other threats. Most of the products on the market are only prototypes, therefore much research and development work as well as standardization have to be conducted to reach market maturity (Gatteschi et al. 2018, p. 9–11).

Opportunities are enormous, mainly regarding the fusion of IoT with blockchain technology. Decision-making could be carried out automatically using smart contracts based on the data received by sensors (Gatteschi et al. 2018, p. 9–11).

Another challenge is the required infrastructure for a successful blockchain adoption. Additional devices for storage and mining are needed. Furthermore, the high data traffic generated by all the components of a smart factory, need a sophisticated communication infrastructure and interfaces supporting this load (Fernández-Caramés and Fraga-Lamas 2019, p. 13–14).

Moreover, standardization is also a challenge regarding blockchain. Currently, most companies develop their own blockchain solution. However, without standardization there is no interoperability between them. IEEE is already working on such standards

driven by the IEEE Standards Association (Fernández-Caramés and Fraga-Lamas 2019, p. 13–14).

There are several challenges regarding legislation when talking about the adoption of blockchain technology. Currently, no adjustments have been made to the law, which would be particularly useful for blockchain and smart contracts (Püttgen and Kaulartz 2017, p. 255). In Austria, the first prenuptial agreement was concluded using a smart contract, however a notary was still needed in order to make the contract lawful (Sara Grasel 2018). Due to the fast pace of information technology, the legislation faces the problem of catching up. This is the reason for the importance of code compliance for smart contracts. Jurists and researchers must work together to reach consensus on how to bring contract clauses into the code of smart contracts. A major challenge will be making rules for undefined legal terms. Nevertheless, this will not pose a problem for the application proposed in this thesis (Püttgen and Kaulartz 2017, p. 255).

In conclusion this topic, the biggest challenge is to find the best solution for the use case at hand. In use cases which only handle a small amount of data, scalability will not play an important role. In other areas such as the smart manufacturing field, scalability will play a significant role, since various systems and sensors will feed data to the blockchain (Panarello et al. 2018, p. 30–31).

In the next chapter the advantages and disadvantages of adapting blockchain technology in smart manufacturing will be discussed.

2.3.2 Advantages & Disadvantages

This chapter concerns the advantages and disadvantages of adapting blockchain in smart manufacturing. Overall, the main advantage is that blockchain enables different entities to reach an agreement on a defined activity and save that agreement without the need for a third party to establish trust. This activity agreed on then gets shared with all other parties of the blockchain (Mohamed and Al-Jaroodi 2019, p. 852).

A few advantages relevant to smart manufacturing can be derived from this principle. The first ones to mention are decentralization and trust. Through consensus algorithms and a trustless peer-to-peer network provided by blockchain technology, smart manufacturing can benefit greatly, having high data replication and no need for a central control unit (Bahga and Madiseti 2016, p. 543–544).

This leads towards a high resilience to failures, as there is no single point of failure, which could be catastrophic for a smart factory. Hackers have only a very slim chance to attack a blockchain network, since the data immutability of information on the blockchain. Once data is stored on the blockchain, it cannot be altered (Bahga and Madiseti 2016, p. 543–544) and (Laabs and Đukanović 2018, p. 144–145).

Scalability is also an advantage in terms of computing power of the network. The more nodes joining the network and users mining, the higher the computing power (Viriyasitavat et al. 2018, p. 4).

One of the most critical advantages of blockchain adaption in the context of smart manufacturing is security. Through blockchain technology strong cryptography is provided, which means all data on the blockchain is highly secure (Viriyasitavat et al. 2018, p. 4). This doesn't make blockchain less transparent, more the opposite. Every user is able to see all transaction on the blockchain and track every process easily, leading to great auditability (Bahga and Madiseti 2016, p. 543–544).

The last major advantage of blockchain adoption is the potential for more autonomy. Through smart contracts many processes can be automated. For example, different IoT devices that communicate with each other and complete transactions autonomously. In this case, every device has its own blockchain account to participate in the network (Bahga and Madiseti 2016, p. 543–544). In Table 15, a mapping of blockchain advantages to demands of smart manufacturing can be found.

Table 15 - Mapping of blockchain advantages to Smart manufacturing demands

Blockchain advantages	Demands of Smart manufacturing
Decentralization	High data replication
Trust	No central control unit – Bottleneck
High resilience to failure	No single point of failure
Data immutability, strong cryptography	Cybersecurity
Scalability	Computing power
Transparency	Auditability
Smart contracts	High Automation

After discussing the benefits an adoption of blockchain can implicate, the disadvantages will be analyzed. The biggest issue worth mentioning is the poor performance of blockchain technology when compared to traditional database systems. The disappointing performance of blockchain is mostly caused by the choice of the consensus algorithm “Proof of Work” and the lack of parallel execution. Adapting the consensus algorithm and changing the size and interval of a block can help with this problem (Laabs and Đukanović 2018, p. 149–151).

Another disadvantage are many small devices with low computing power in the environment of Industry 4.0. This could lead to problems with larger blockchains because calculations would take longer on such devices.

Key management can also become an issue. This is not only true for blockchain technology, but also counts for other systems with the need of keys. A type of system needs to be installed to manage these keys, since if an attacker gets a key, he has access to the blockchain and every security measure of blockchain fails (Laabs and Đukanović 2018, p. 149–151).

Lastly, privacy can be an issue with blockchain, since they only provide pseudonymity and not anonymity. In the context of smart manufacturing this only plays a minor role because business partners will know each another. However, it is possible to keep some transactions between partners private. This is also a problem with the well-known public blockchains such as Bitcoin and Ethereum, but there are already solutions like the Zerocash protocol (Laabs and Đukanović 2018, p. 149–151). Table 16 shows a mapping of blockchain disadvantages to demands of smart manufacturing.

Table 16 - Mapping of blockchain disadvantages to Smart manufacturing demands

Blockchain disadvantages	Demands of Smart manufacturing
Blockchain versus traditional databases	Fast computing
Disk space of lager blockchains	Small IoT devices
	Key management
Transparency versus Privacy	Privacy

Finishing off this chapter, it is important to note that the design and architecture of the overall system is the key to success, it comes and goes with the advantages and disadvantages mentioned in this section. The decision starts with whether a public or consortium type blockchain is better suited and continues with the consensus algorithm as well as surrounding supporting systems. Such a system could be an additional technology for decentralization, since blockchain is not well suited to save big amounts of data. An additional data layer eliminates the problem that blockchain only saves the hash of the place where the data is stored (Laabs and Đukanović 2018, p. 149–151).

In the following chapter technological and non-technological risks will be analyzed.

2.3.3 Technological and Non-Technological Risks

With the promise of reduced cost and increased efficiency when using blockchain technology also come some risks. Overall two types of blockchain must be

differentiated, namely permissionless and permissioned blockchain (Zamani et al. 2018, p. 4–5).

Permissionless blockchains such as bitcoin have an open source code. It means that everyone can examine the code, which can be advantageous, but is also a risk when unknown security vulnerabilities are exploited (Zamani et al. 2018, p. 4–5).

Another major risk are so-called crypto wallets, in which users save their private keys. In the industrial context, it does not need to be a crypto wallet, but this private key has to be stored somewhere. If a user has this private key, he can do everything with the account associated with it, no matter whether he is actually the owner (Zamani et al. 2018, p. 4–5).

Permissioned blockchains grant a more controlled environment because governing parties are in control. They are able to choose trusted nodes and give them access to the blockchain. An associated risk is granting incorrect permissions. Since such a network is usually smaller than a public one, a few malicious nodes could cause a significant amount of damage (Zamani et al. 2018, p. 5–6).

Another risk for permissioned blockchain is interoperability, due to every private blockchain running on different data models and permission controls. The option to connect two private blockchains would be much work. Additionally, the use of third-party applications for permissioned blockchain surfaces is a great risk. Not being in control of this applications possibly means threats to security, because the security is only as strong as its weakest link (Zamani et al. 2018, p. 5–6).

Lastly, a technological risk is the low standardization of the blockchain environment. The development of blockchain is not very advanced and a blockchain often is not the only technology used but is connected to IoT devices or a cloud, which increases the security risk (Zamani et al. 2018, p. 6–7).

Non-technological risks are mainly associated with people who do not know what blockchain is and does, which can lead to skepticism and resistance. Also, it is a major financial risk. The future of blockchain and its role is uncertain and investing in it means investing into a great amount of research and development that has to be done (Bizarro et al., p. 12–16). In Table 17 a summary of all risks can be found.

After having discussed the advantages, disadvantages, risks and challenges of smart manufacturing and blockchain, Figure 12 shows a Venn diagram, which links all these factors. Scalability, Privacy and the secure data collection for instance, are challenges, risks and advantages or disadvantages.

In the next chapter, possible use cases for the adaption of blockchain in the smart manufacturing environment will be discussed.

Table 17 - Technological and non-technological risks

Risk	Type of risk
Technological risks	open source code- vulnerabilities, exploits
	security depends on securely kept private key
	incorrect permissions
	missing interoperability
	Use of third-party applications
	Low standardization
Non-technological risks	Financial risk
	Unknown future
	Uncertainness
	Low understanding

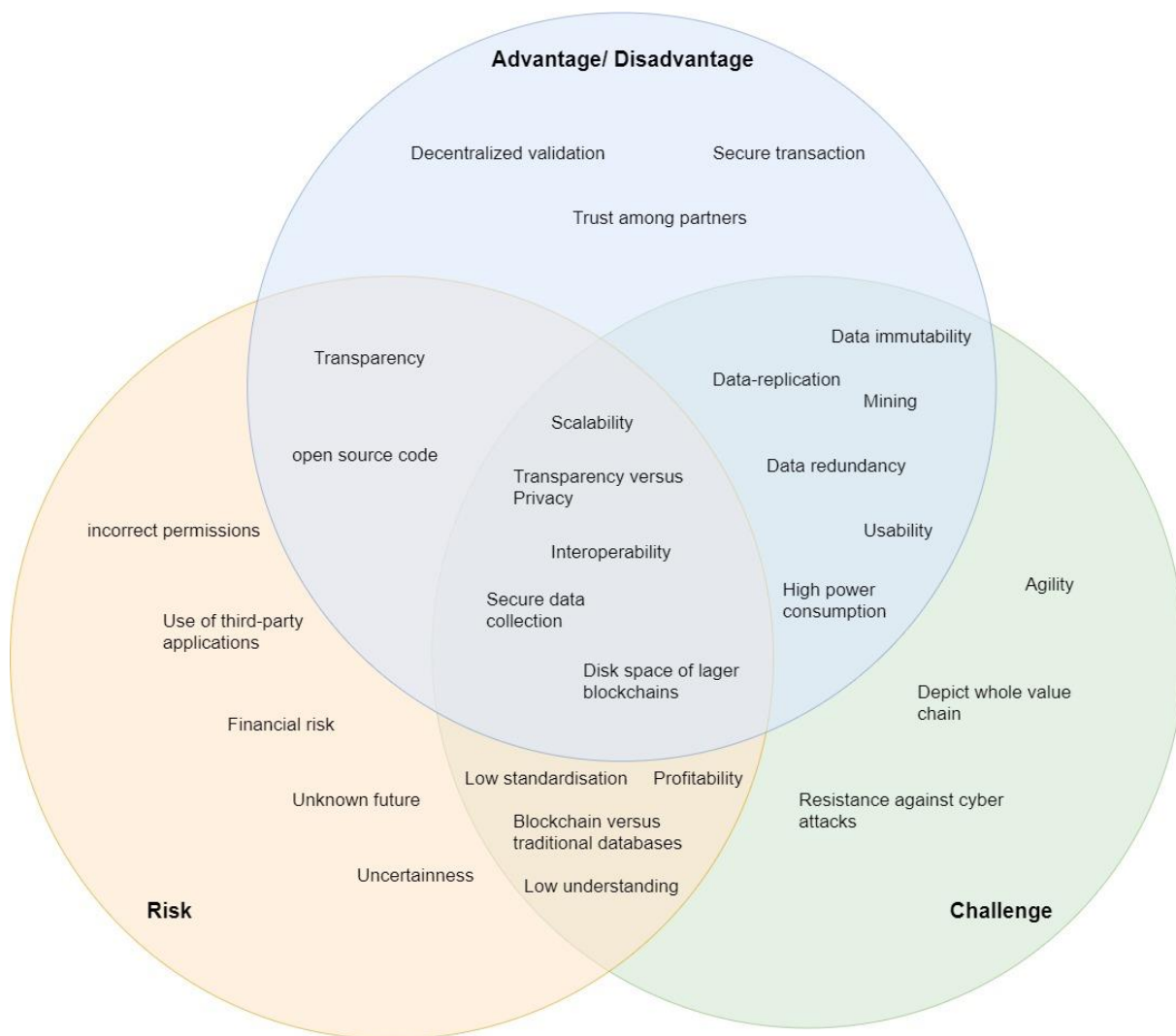


Figure 12 - Linking of Advantages, Disadvantages, Risks and Challenges

2.3.4 Use-cases

After having analyzed the challenges, advantages and disadvantages of adopting blockchain in smart manufacturing, some possible use cases are discussed.

One possible use case is the identification of IoT devices. This could solve the problem of identifying IoT devices. Currently certificates are used, which is costly and also not very scalable. Blockchain fits well for this, since it would greatly reduce vulnerability during the registration process. The device could be registered with a digital identity on the blockchain, which cannot be manipulated after its creation. In addition, information about the device can be dynamically updated, which is not possible when using certificates (Dieterich et al., p. 6–9).

Another use case is to timestamp data from sensors to prohibit manipulation. The idea is to not only save the data from sensors on the blockchain, but also save the timestamp. This increases the trust between different parties, since it can be ensured that data has not been altered after the creation (Dieterich et al., p. 6–9).

The next possible use case is a marketplace for suppliers to sell data from IoT devices. The suppliers could save the data generated from their IoT devices and give customers access to their blockchain. Customers could gain access to real time data and also pay using the blockchain (Dieterich et al., p. 6–9).

A protocol developed for demands of IoT devices is another use case. A way of handling downsides such as problems with scalability, low verification speed and transaction fees is needed. An algorithm operating in a manner that every user is required to work to be able to make transactions could reduce the problems with transactions fees and scalability (Dieterich et al., p. 6–9).

Another very interesting use-case would be a platform to save data from IoT devices on a private blockchain and share it with business partners. Every participant of the private blockchain can share their data on the private blockchain in a safe manner (Dieterich et al., p. 6–9).

The last use case mentioned in this thesis is a scenario where there are a company producing parts in a smart factory, another company responsible for maintenance and one original equipment manufacturer. These three companies are distributed across the globe. A blockchain could connect these three companies in a way that the producing company has an overview of its machines and their statuses, the OEM gets orders for spare parts and the maintenance company has access to all machine data to detect possible future problems.

Thus, a big part of the supply chain is feeding data into the blockchain and all three parties have access to the necessary data. This way the original equipment manufacturer can react faster and the producing company does not have additional

expenditure. This use case could solve an important challenge in Industry 4.0, which is the cooperation among stakeholders throughout the value chain (Thames and Schaefer 2017, p. 1–2). Lastly, a summary of all use cases and the context to smart manufacturing can be found in Table 18.

Table 18 - Use-cases and possible application areas in smart manufacturing

Use-case	Smart manufacturing problem areas
Identification of IoT devices	Security
Timestamp data from sensors	Maintenance, Security
Marketplace for suppliers	Sales
Protocol for IoT devices	Security
Platform to save data form IoT devices	Maintenance, Security
Data sharing platform	Maintenance, Logistics

3 Security in Manufacturing

In this chapter, the state-of-the-art of security in smart manufacturing will be reviewed and analyzed. First, existing solutions in the context of smart manufacturing will be discussed, then solutions in other fields will be evaluated.

3.1 Cybersecurity in Industry 4.0

In Industry 4.0, where the virtual and physical world become more and more integrated, security as well as privacy turn into important issues due to the distributed nature of such systems and real time data acquisition. Existing solutions for other fields might not be applicable for the industry sector, since other requirements and rules are at place. For example, looking at security for IoT in smart manufacturing, there are a lot of distributed components like sensors with small computing power, which need protection. Huge amounts of data are collected, which is why privacy plays a significant role (Xu et al. 2018, p. 2944–2945).

Industrial IoT is a subset of IoT, which contains embedded computing and communication technology. These systems focus mainly on sensor technology and the collection as well as transmission of data, in which communication plays an essential role. Industrial IoT uses wired and wireless communication technology. Examples of protocols used are: Ethernet, Wi-Fi, 2G/3G/4G, HTTP, MQTT and many others. MQTT is a public subscribe messaging protocol, which was specifically designed for industrial network environments (Thames and Schaefer 2017, p. 3–4). One security challenge arising is the low computing power of IoT devices, which leads to not being able to perform complex cryptographic operations. Another challenge is the exposedness of IoT devices to potential malicious users. Lastly, the fact that a connection to an IoT device may not always be possible if, for instance, the device goes to “sleep mode” to save energy (Polyzos and Fotiou 2017, p. 75).

An interesting model of smart manufacturing and Industry 4.0 is the cloud-based design or cloud-based manufacturing. Cloud-Based Design and Manufacturing (CBDM) is a product realization model, which supports innovation and rapid product development. It is a distributed and parallel system, which consists of physical and virtual components. An overview of such a CBDM concept can be seen in Figure 13. In the center, the cloud resources can be seen with all the connections to virtual resources, physical resources, product development and realization, human resources and others (Thames and Schaefer 2017, p. 11–12).

Following are the most important characteristics of CBDM:

- a) **Scalability:** With CBDM systems rapid scalability is possible. Machine parts, machine tools, material handling units, personnel and other components can be

- modified, added or removed easily. This leads to having more dynamic capacity planning and being able to handle transient demand (Wu et al. 2012, p. 320).
- b) **On demand self-service:** With the cloud a customer or anyone with access to the cloud can provide or buy engineering resources and manufacturing hardware on demand (Wu et al. 2012, p. 320).
 - c) **Resource pooling:** The components and resources used by a manufacturer can be served in a pay-per-use fashion. The CBDM provides access to an on-demand pool of configurable manufacturing resources. Using smart technology, the right amount of a resource can be calculated, which ensures an efficient cloud resource allocation (Thames and Schaefer 2017, p. 12–13).
 - d) **Network access:** Omnipresent network access is needed for the so-called customer co-creation paradigm, which enables the proactive interaction with customers. Global network access is needed in order to easily reach the needed level of communication. CBDM allows stakeholders to participate in the entire production cycle (Thames and Schaefer 2017, p. 12–13).
 - e) **Virtualization:** CBDM provides a virtual environment through simulation, which enables companies to independently view software packages computing and data storage resources from hardware (Bohn et al. 2011).

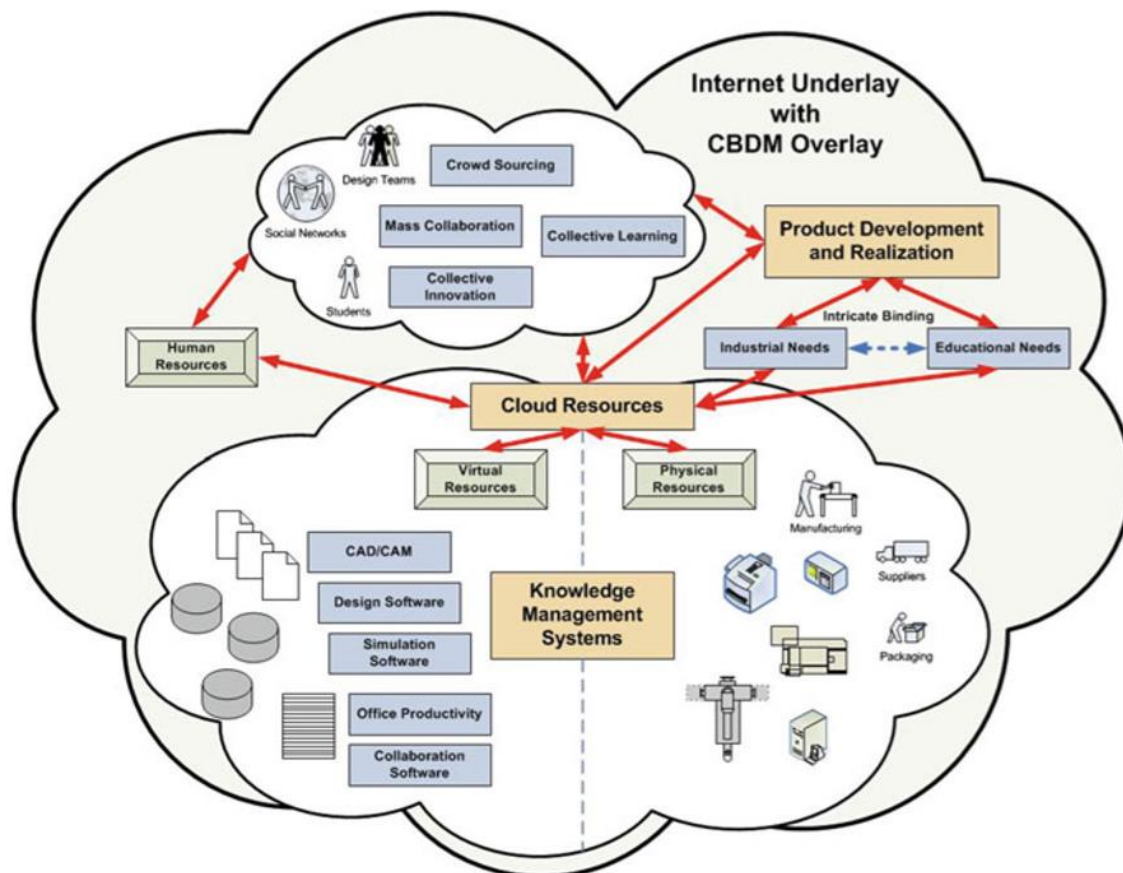


Figure 13 - Cloud-Based Design and Manufacturing (Thames and Schaefer 2017, p. 12)

Diving deeper into the concept of CBDM, Figure 14 depicts a distributed infrastructure with centralized interfacing system model. Components of these models are: user interface components, which are, for instance, web browsers communication and security components, which are the internet and firewalls, stakeholders, which can be users, consumers, managers and others, and the manufacturing process assets, which are software components, CAD tools, as well as physical components like 3D printers (Thames and Schaefer 2017, p. 18–19).

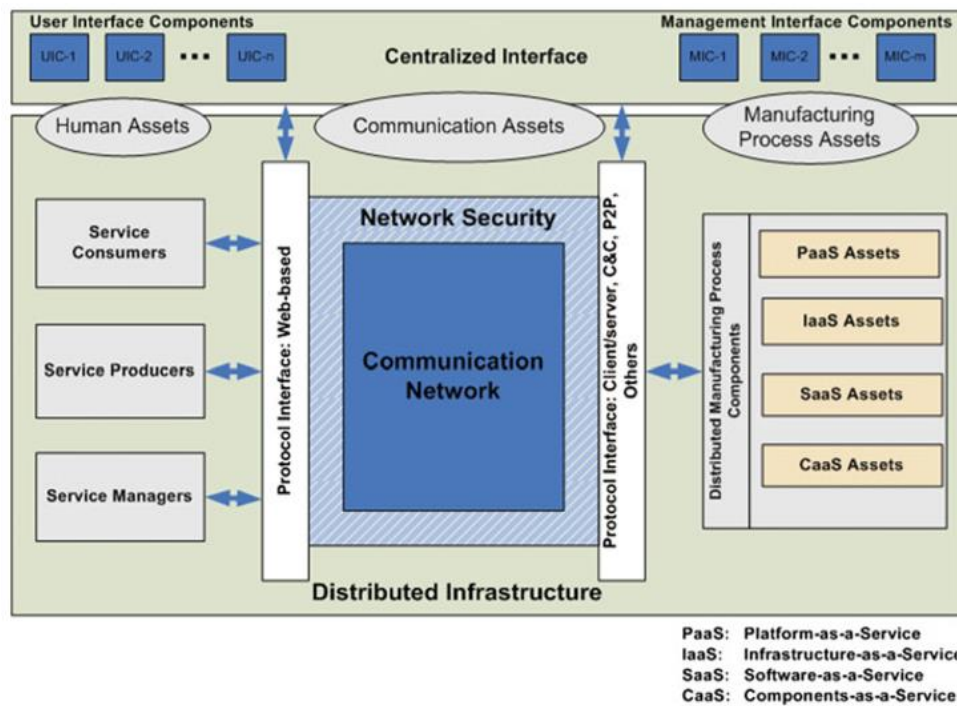


Figure 14 - Distributed infrastructure with centralized interfacing system (Thames and Schaefer 2017, p. 19)

Concerning the distributed manufacturing process components, the following services can be differentiated (Wu et al. 2014, p. 2–3):

- Platform-as-a-Service (PaaS):** The goal of PaaS is to provide an environment and a set of tools that help with the integration of required functionality. As an example, Fujitsu offers a platform for a high speed thin client that reduces manufacturing costs and development times.
- Infrastructure-as-a-Service (IaaS):** IaaS grants computing resources such as storage space or servers on a pay-as-you-go basis, which eliminates downtimes and reduces costs.
- Software-as-a-Service (SaaS):** SaaS provides software applications such as CAD or enterprise resource planning software. The software can be installed and also run through a thin client interface without purchasing a full software license. An example would be Dassault Systems that provide remotely running 3D software environments.

- d) **Hardware-as-a Service (HaaS):** HaaS provides a customer with additional hardware environment, which could be machine tools, or manufacturing process related.

Lastly, there is a software defined cloud manufacturing architecture that enables cyber physical product creation. The architecture of such a system can be seen in Figure 15. This structure is assumed to be used in an environment with a large network of software and hardware components with internet-based communication. Since the separation of concerns is an important concept, this architecture is separated into the software plane and the hardware plane. Due to separation, it is possible to distinct between hardware elements and software elements. On the hardware plane, the distributed hardware layer and several distributed hardware elements can be found. A hardware element could, for instance, be a 3D printer. On the software plane there is the virtual layer with final user applications and the control layer with control elements. The communication takes place via a communication interface (Thames and Schaefer 2017, p. 25–27).

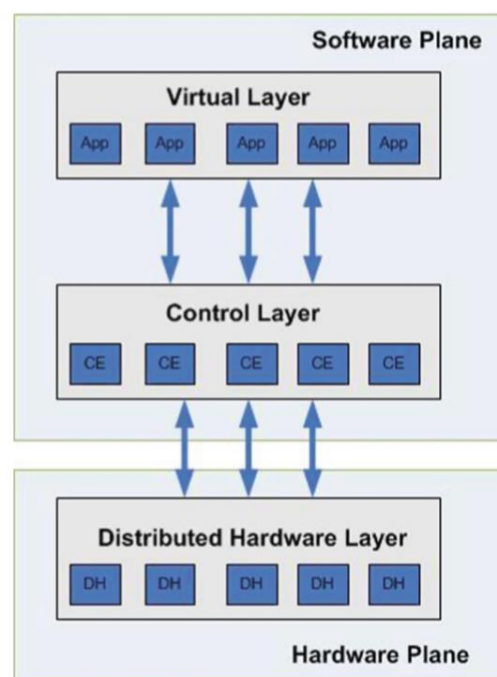


Figure 15 - Software defined cloud manufacturing architecture (Thames and Schaefer 2017, p. 26)

In conclusion, one challenge of security for CBDM seems potent. The focus of security cannot only lay on information technology because operational technology plays an important role as well in the context of smart manufacturing. Chapter 3.1.2 analyzes this in more detail. In the following chapter, the encryption of CAD models in a cloud-based system will be discussed.

3.1.1 Encryption of Computer Aided Design Models

Collaboration or sharing information with a partner is a trend in Industry 4.0. It is used either to optimize designs or to keep a business partner up to date. In this chapter encryption of sensitive data will be analyzed on the example of so-called Computer Aided Design (CAD) models. Three features are necessary in order for everyone to receive the required information (Thames and Schaefer 2017, p. 36–37) and (Ouyang et al. 2004, p. 581–583):

- Public features do not contain any sensitive data therefore, this information can be shared with collaborators without deformation. An example is the basic CAD model in Figure 16.
- Shared features are distributed among collaborators and encrypted by the owner and decrypted by the collaborator. An example is detailed information on the hole in Figure 16.
- Protected features are not shared with anyone and only visible by the owner of the CAD model. The blades of the model in Figure 16 is an example of the collaborator being able to see the blades but no information beyond that.

In Figure 16, the process of sharing a CAD model is depicted, in which the owner builds a model and encrypts it. The shared features are decrypted by collaborators.

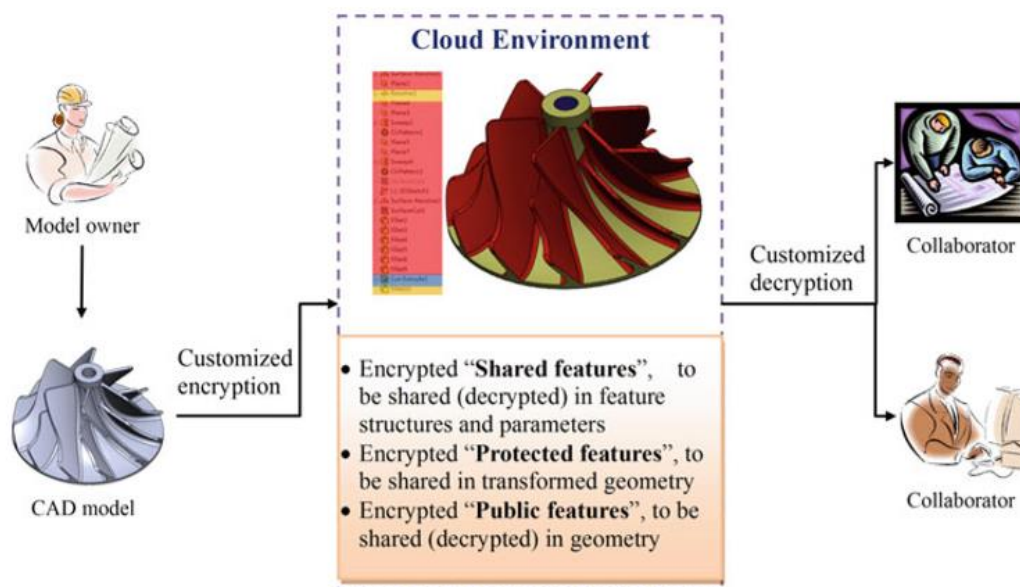


Figure 16 - Example of collaboration on a CAD model (Thames and Schaefer 2017, p. 36)

After getting a basic idea about this use case, Table 19 shows different security aspects for collaborating with business partners. The first security measure that can be applied is the use of watermarks, which provides information authentication as well as data level security. In the CAD model, the creator embeds their personal watermark, which cannot be removed by the collaborators but can be detected by software.

However, this approach cannot handle data hiding (Thames and Schaefer 2017, p. 35–38).

A method for securing data and having an architecture level security is to use access control. There are four different approaches, namely the general access control, access control for files, sharing space access control and the multi method control (Wu et al. 2016, p. 466). The general approach was first mentioned in the 1970s along with the concept of access matrices. The file-based approach is useful since the access to data is still file-based. The sharing approach uses a sharing space where collaboration takes place. Lastly, the multi method approach combines different security mechanisms to ensure safety (Thames and Schaefer 2017, p. 39–40).

Table 19 - Security for collaboration (Thames and Schaefer 2017, p. 38) and (Wu et al. 2016, p. 466)

	A	B	C	D	Purpose	Problems
Watermark	X			X	Protect the intellectual property by embedding the watermark into CAD models	Design information (design knowledge, parameters) cannot be hidden by watermark in a safe way
Access control		X	X		Control the access of the design data by users' authorization according to a group of access control rules	Unable to support information protection of a CAD model
Multi-resolution approach		X		X	Multi-resolution is to simplify a CAD model during data sharing in a network with limited bandwidth	It can realize the secure sharing of a CAD model to some degrees, but not flexibly
Encryption of CAD models		X		X	Hide the design information by encryption	No research applied to CAD models
Notes: A--Information authentication; B--Information hiding; C--Architecture-level security; D--Data-level security						

The next chapter will analyze the Cyberphysical security in the smart manufacturing environment and show different architectures as well as solution characteristics.

3.1.2 Cyberphysical Security

One important difference of security in the context of Industry 4.0 is that not only information technology has to be protected, but also operation technology. Paradigms used for information technology are not necessary to use for operation technology, which is often the case in the real world. There are some critical types of data in operation technology to protect, namely bill of materials, design information and control

parameters. A practical example would be a 3D printer, which delivers design information and control parameters to an operator. The operator processes the data and sends it to the controller boards operating the printer. The goal of a secure approach is to not give critical information to any person or device other than the lowest level controller (Thames and Schaefer 2017, p. 59–61). A demarcation of different security concepts from the European Union Agency for Network and Information Security can be found in Table 20.

Table 20 - Security concepts (Moon et al. 2018, p. 6)

Security concept	Description
Information security	Protection against theft, deletion or alteration of stored or transmitted data in a cyber system.
Operations security	Protection against malicious changes of practices or workflows.
Communications security	Protection against an attack on the technical infrastructure of a cyber system, which changes its behaviour leading to malicious activities.
Physical security	Protection against physical threats, which influence or damage the correct behaviour of a cyber system.

When talking about security there are three main pillars: confidentiality, integrity and availability or in short CIA. The objective of cybersecurity is to carefully balance these three in order to reach a secure solution. Traditional security approaches focused on layered defenses around a central core server, not putting too much effort into protecting peripheral devices such as a printer. But what happens if the printer gets hacked? How much damage will be done? Through the emergence of smart manufacturing and IoT this paradigm must shift. The central devices and all remote manufacturing equipment, such as sensors and 3D printers, must be protected (Bishop 2015).

Protection can be provided by keeping in mind the CIA objectives. Confidentiality can be ensured by keeping the manufacturing data secure. Data regarding the design also must have integrity therefore, the information should not be altered. Lastly, the data needs to be available to send it to a 3D printer and ,for instance, build equipment (Thames and Schaefer 2017, p. 59–61) and (Bishop 2015).

A shift of a central to a distributed security paradigm can be observed, which requires trust through the whole value chain providing confidential and not altered information when it is needed. To ensure this extensive monitoring, a control based security is needed (Behl and Behl 2012, p. 200–202).

A main issue regarding security in Industry 4.0 is the problem of authentication. Who is authorized to do what? There are two concepts to solve this problem, namely asymmetric encryption keys and comptrollers. Asymmetric encryption keys consist of a so-called public key and a private key. The public key can be used by a person or a piece of software to encrypt data. The receiver of information can decrypt this information with the right private key. If this process is successful, the receiver is authorized. A comptroller has the function of taking some input data, distributing a key and storing output data, as can be seen in Figure 17 (Thames and Schaefer 2017, p. 63–64) and (Behl and Behl 2012, p. 201–203).

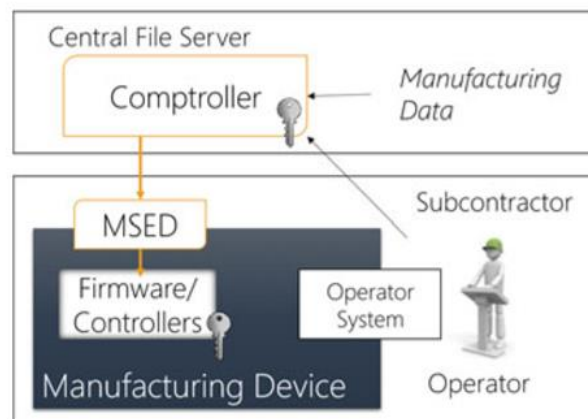


Figure 17 - Manufacturing workflow (Thames and Schaefer 2017, p. 63)

The disadvantages of a central authentication and authorization entity seem obvious. A single point of failure is created, which can be used by hackers, becoming the single primary target (Thames and Schaefer 2017, p. 66).

Concludingly, a list of critical solution characteristics (Thames and Schaefer 2017, p. 69):

- a) Granular authorization allows for different levels of permissions such as viewing or editing rights.
- b) Distributed responsibility allows for a dispersed security risk instead of a single point of failure.
- c) Operation control can be maintained, monitored and restricted if necessary.
- d) Device support is given due to an operating client.
- e) The solution is location independent and can be deployed central or hosted.
- f) Only necessary data is transferred.

If these characteristics are met by a security solution, then tracking details is possible due to the comptroller, collaboration is encouraged, distributed manufacturing is enabled and a central processing unit improves automation (Thames and Schaefer 2017, p. 70).

3.1.3 Threats

Cyberattacks such as denial of service, man in the middle or cross site scripting attacks, create a Tier 1 risk. This is the reason security improves trust and thereby the willingness to collaborate and the competitive advantage. With Industry 4.0 hinging on Cyber-Physical Systems, IoT and IoS, the main security focus lies on communication (Thames and Schaefer 2017, p. 226–227) and (Tuptuk and Hailes 2018, p. 98).

The most common results of such security incidents are according to TÜV (TÜV AUSTRIA and Fraunhofer Austria 2018, p. 10) the theft of licenses, compromising of E-mails, theft of patent law, financial loss and planted ransomware.

Security in IoT can be divided regarding the fundamental IoT architecture, communication threats, IoT applications, or threats coming from human users. Furthermore, they can be separated into logical meaning decision making or use of data, software and hardware threats (Thames and Schaefer 2017, p. 229).

An attack can be caused by humans, due to technical insufficiencies or by targeting the actual hardware. Examples for security issues caused by humans involuntarily or deliberately can be (Thames and Schaefer 2017, p. 230–233):

- Wrong data entries
- Improper use/disposal of sensitive data
- Improper use of electronic equipment
- Ignorance of errors or warnings
- Insufficient password management
- Procedural violation
- High level data analytics

Threats from internal technical issues regarding software or hardware can be (Thames and Schaefer 2017, p. 230–233):

- Corruption by system errors
- Insertion of malicious software
- Poor programming style
- Insufficient authentication method
- Power failure
- Poor communication protocols

Examples for threats coming from communication through physical means are (Thames and Schaefer 2017, p. 230–233):

- Physical tampering of hardware
- Electromagnetic attacks
- Introduction of hazardous materials

- Mechanical attacks such as cutting wires
- Side channel attacks such as timing attacks or power analysis attacks

These examples illustrate the threats of the IoT infrastructure arising from special circumstances such as the embeddedness and minimal maintenance of IoT devices as well as the constant need of a power supply, limited memory size and the problem of updating such devices.

Additionally, new Quality Control (QC) systems are needed, since in the early 2020s such systems mainly focus on possible quality losses but not on attacks on a CPS. Current QC systems are not designed to detect changes to a production process caused by a malicious attack. In 2015, the number of days between an attack and its detection was 200 days. Also, 69% of breaches were not detected by the involved party but instead were detected by other parties such as law enforcement. Now the question arising is what would happen to a company that produces faulty parts for 200 days (Elhabashy et al. 2019, p. 2489–2493)?

Finishing of this chapter, Table 21 gives an overview of the areas a company should focus their attention on in terms of securing a system in Industry 4.0.

Table 21 - Focus areas of Security (Thames and Schaefer 2017, p. 192)

Access Control	Separation of Duties	Intrusion Detection Systems	Adversary and Trust models
Audit	Timestamps	Security Management Process	Redundancy
Removal of unneeded applications	Authentication	Architecture	Defense in Depth
	Secure Communication Infrastructure	Digital Signatures	

3.1.4 SCADA Systems

Supervisory Control and Data Acquisition (SCADA) is a control system architecture used in manufacturing. SCADA systems are widely distributed systems that gather and analyze real time data from devices. They control several Programmable Logical Controllers (PLC) and Remote Terminal Units (RTU), which uses sensors to gather data. Originally, SCADA systems were designed to work in closed networks only focusing on keeping intruders outside but neglecting other security aspects (Kruz 2006, p. 3–7).

A conceptual architecture of a SCADA system can be seen in Figure 18 with two main sections being the control center and the field sites. SCADA hardware consists of programmable logic controllers and remote terminal units, which are computerized devices that are connected to sensors and control automated processes. Intelligent electronic devices are used to monitor, control and communicate independently. A control center is the receiver of data and is also part of SCADA hardware. Lastly, Human Machine Interface (HMI) is to mention, which has the objective to interpret and visualize data (Thames and Schaefer 2017, p. 78–79).

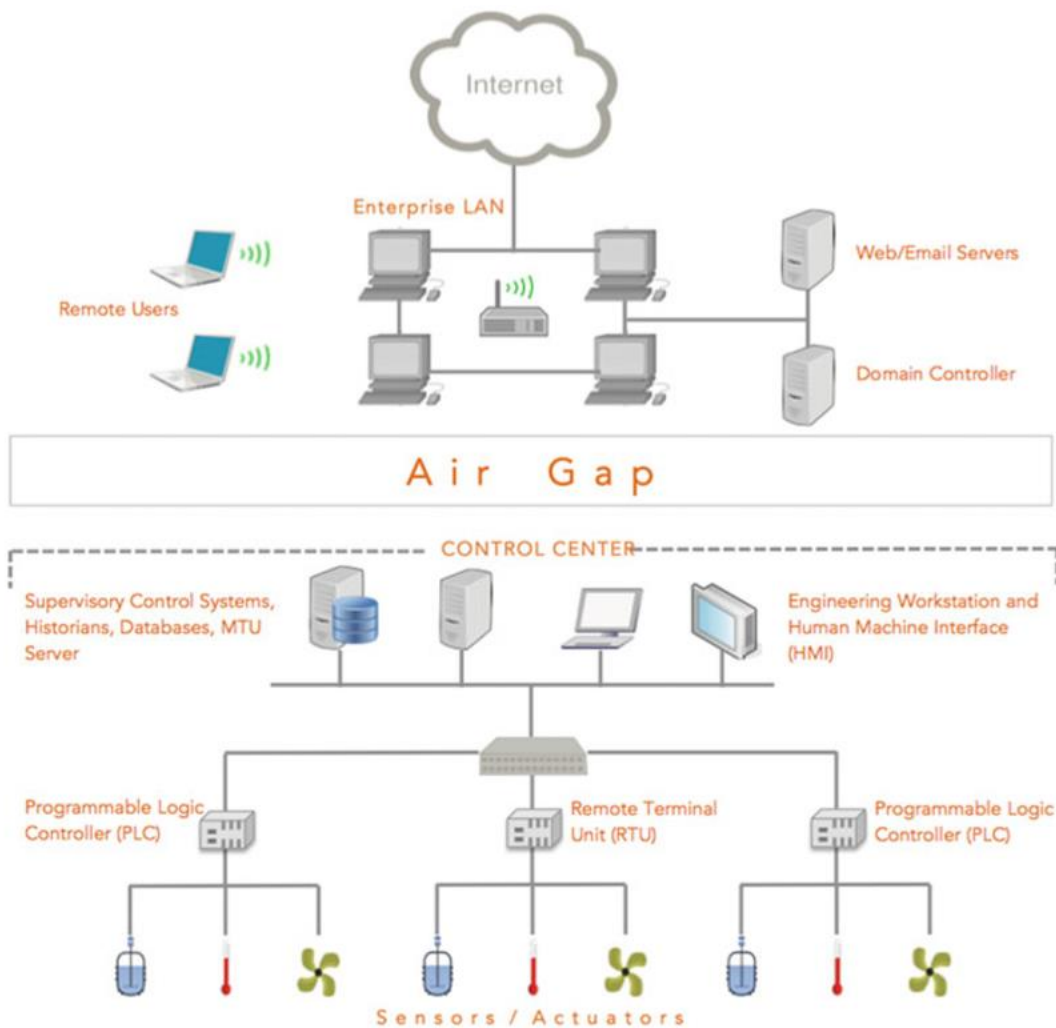


Figure 18 - SCADA architecture (Thames and Schaefer 2017, p. 78)

SCADA software can differ depending on the objective and use-case, but the networking behind the system is important for security. Throughout the years, SCADA systems developed from a closed approach to relying more and more on cloud-based solutions, which offer a broader level of control and monitoring from outside. However, the downside to a more open approach is the increased security risk, since intrusion is easier.

The network structure of a modern SCADA system which is operation in the IIoT space, can be divided into different zones. This is shown in Figure 19. The goal of this approach is to increase security by performing the most critical communication in the lowest and also most secure layer (Ahmed et al. 2012, p. 45–47).

The **control zone** consists of three layers. In layer 0, the field devices consisting of PLCs and RTUs are connected over a bus network. In layer 1, there are controllers which receive signals from field devices over an electrical input. These can be decoded under the use of network protocols and outputs can be sent back. The last layer in the control zone is layer 2, which is connected to layer 1 and receives information from the lower layers. This information is handed to a HMI to be interpreted and controlled (Thames and Schaefer 2017, p. 80–82).

The **data zone** consists of layer 3 and of historians, which are database management systems that store data to carry out forensic investigation, application servers and domain servers. This can be seen in Figure 19 (Ahmed et al. 2012, p. 45–47).

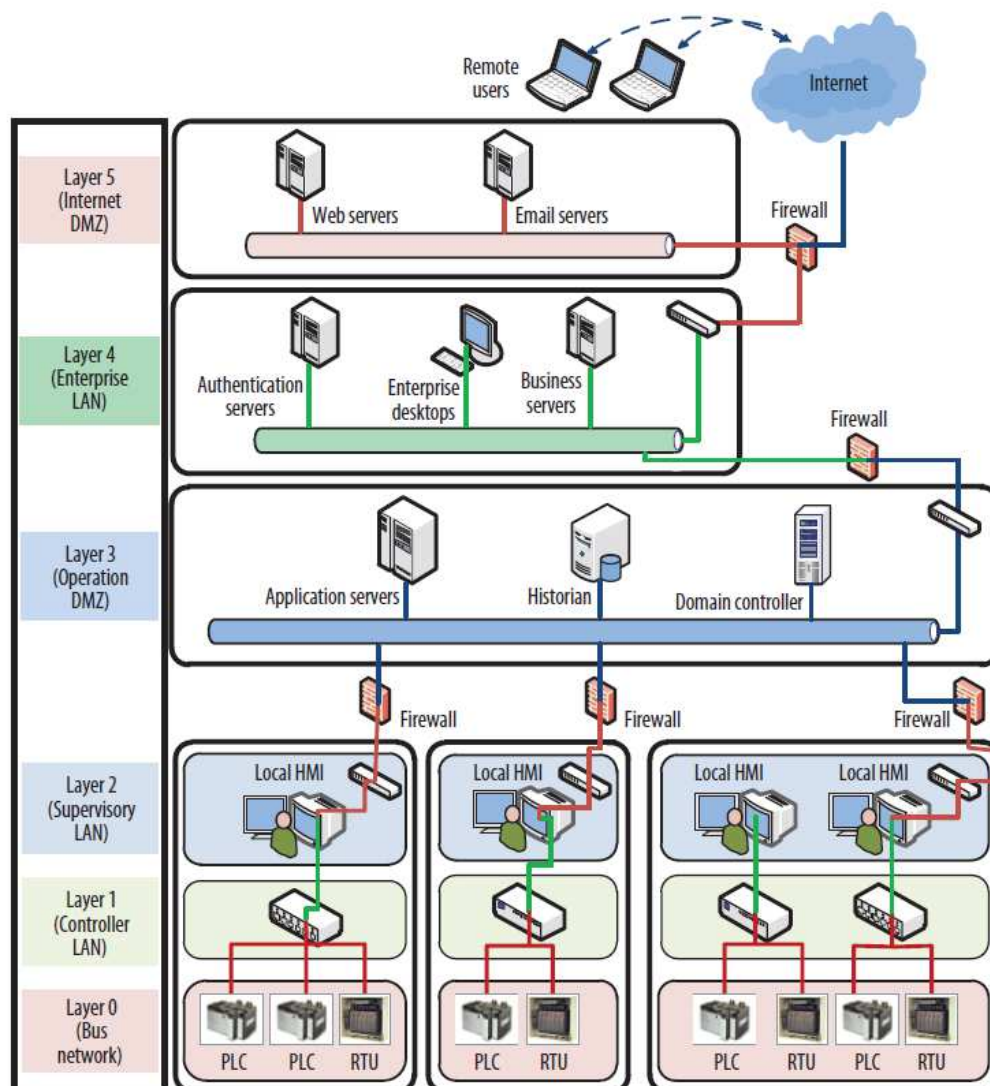


Figure 19 - Communication zones of a SCADA system (Ahmed et al. 2012, p. 46)

The **corporate zone** consists of layer 4 with all enterprise servers used for E-Mail, and other business workstations needed for company communication (Ahmed et al. 2012, p. 45–47).

Lastly the **external zone** consists of layer 5 and connects remote operations and third-party vendors (Ahmed et al. 2012, p. 45–47).

Having discussed the architecture of SCADA systems on different levels, it is important to have a backup plan, in case of a breach. Therefore, so-called forensic responses to a SCADA system failure are needed to identify the origin of the incident. The knowledge whether the system is still at risk and the identification of damages as well as an analysis of the weakness of the system are critical (Ahmed et al. 2012, p. 45).

Due to the high complexity of SCADA systems, live forensics are essential to continuously being able to stay operational. Through live forensics it is possible to acquire and analyze data while the SCADA system is running. Another critical factor in case of an incident is a rapid response. Evidence left on physical memory will decrease over time when it is overwritten. This might sound simple, but the geographical size of SCADA systems presents a real challenge in that regard. Obtaining evidence is also an important part of forensics, since the integrity and validity of data has to be ensured in case of a business having to answer to a court of law (Thames and Schaefer 2017, p. 85–88) and (Ahmed et al. 2012, p. 45).

Lastly, the forensic methodology for SCADA is important to understand. There are six stages involved. The first stage is preparation, where the systems architecture and requirements have to be understood in order to gain insights on possible attacks. The second stage is detection, in which the type of attack and the infected areas have to be determined. In the third stage, the infected areas have to be isolated. In the next stage, which is called triage, the data sources need to be identified and prioritized. In the last two stages, data analysis and acquisition are performed and a report is being created on what needs to be updated as well as all other findings (Thames and Schaefer 2017, p. 94–97).

In conclusion, forensics play an important role when facing security challenges. Through live forensics and a rapid response, integrity and validity can be ensured. In the next chapter the topic of big data will be discussed, since it is a more and more relevant topic due to the rising amounts of data being generated.

3.1.5 Security in IoT

Before analyzing the role of big data in Industry 4.0 and smart manufacturing, it is important to know the three types of integration into a system based on Industry 4.0. The first type is horizontal integration, which is used to define an efficient ecosystem that allows for data to be shared among business partners. The second type is vertical

integration. It has the goal of achieving elastic and reconfigurable manufacturing systems. The best example, in which vertical integration is used, are actuators and sensors. They are connected over all levels of the system, which enables a form of self-organization. The last type is End-To-End engineering integration, which includes activities such as the design and development of manufactured goods, planning and production processes, inventions and other services. The goal is to enable reusability of consistent product models (Thames and Schaefer 2017, p. 104–105) and (Lee and Seshia 2017, p. 200–202).

After having discussed the different types of integration, it is important to understand the different layers starting with the application layer, hardware layer, embedded layer, communication layer, secure layer, integration layer and database layer. This architecture ensures efficient communication between components on the internet. The tasks and possible technologies behind that can be seen in Table 22 (Thames and Schaefer 2017, p. 107–111).

Table 22 - IoT layers (Thames and Schaefer 2017, p. 111)

IoT layers	IoT components	Tasks	Technologies
Application layer	Applications	Provide security against viewing data	Smart home technology, robotics, cloud computing, fog computing
Hardware layer	Device discovery, access control and data management	Enables communication between applications and components	CoAP, MQTT, REST, OMA Lightweight, OMA DM, EPC, ONS
Embedded layer/sensing layer	Physical objects	Collect, monitor, identify, and provide data about disabled users	RFID, sensors, actuators
Communication layer/network layer	Communication technologies	Wireless WAN: Transmit information over Internet from devices Wireless PAN/LAN: Enables devices to share information	Wireless WAN: 2G, 3G, Long Term Evaluation (LTE), Long Term Evaluation-Advanced (LTE-A), 4G, 5G, Satellite networks, etc.
Secure layer	Embedded security, application security	Securing the components which are connected by internet, applications deployed in IoT	PKI Certificate, encryption and decryption technologies, cryptography tools

IoT layers	IoT components	Tasks	Technologies
Integration layer	Hardware layer to fog to cloud integration, devices to fog server integration	Communication from devices to fog server and fog to cloud remote servers	Java Web services, AWS
DB layer	Database technologies	Connecting the applications to data base in the cloud and fog	Oracle Cloud, Microsoft Azure, AWS EBS, AWS EMR,

This layered architecture in IoT entails vulnerabilities and opens possibilities for attacks. In Table 23, these can be found itemized by different components, being physical objects, communication technologies and applications (Thames and Schaefer 2017, p. 112–113).

Table 23 - Security vulnerabilities and solutions of IoT (Thames and Schaefer 2017, p. 113)

Components	Vulnerabilities	Types of threats and attacks	Security solutions
Physical objects	limited communication, calculation and storage resources, distributed in various regions → unauthorized user can access devices and can do harm	DoS/DDoS attacks, Physical attacks, Integrating WSNs, Integrating RFID, Unauthorized access control and data access	Encryption/Cryptographic techniques, Continuously evaluation of suspicious nodes behaviour, Authentication, Authorization, Access control, Identification
Communication technologies	IoT is a dynamic network infrastructure, Power issues, Network issues, Selection of security technique and its challenges	Wireless WAN communications, Wireless LAN/PAN communications, Secure IoT communication protocols in constrained resources environment, Secure transmitted data	Communication security, Encryption/decryption, Strong authentication, Backup solution in case of failure IoT communication protocols enhancement Authorized access and availability
Applications	Data coverage, Cloud	DoS, XSS attack, CSRF attack, SQL	Data separation between information content and

Components	Vulnerabilities	Types of threats and attacks	Security solutions
	computing, Security issues in web application, Secure communication	Injection, Data protection, Data access, PHRs Attacks, Malicious user attacks, Sharing data in different environments, Real-time information processing, Sharing same sensed data by several applications	information source, Encryption/decryption, Secure data access, Confidentiality of data, Secure sensitive data, Backup plan, Traditional distributed database technology, Scheduling techniques, Assuring identification, Assuring authentication, Firewall and antivirus, Intrusion detection, Enhanced communication protocols security

After having discussed different IoT layers and vulnerabilities, a meta cloud data storage architecture is proposed. A meta cloud is used to process and store different applications, which are deployed in the cloud provider, because security to big data is of high importance. The main function of this architecture is to collect data generated from cloud data centers, to process data from IoT devices and to use this data to be able to make optimal decisions. This architecture ensures protection to the big data process and can be seen in Figure 20 (Thames and Schaefer 2017, p. 114–120).

Data is stored on cloud storages from Amazon or Google. This data is classified into three levels: sensitive, critical and normal. Each of those types is stored in a different data center. An interface is used to redirect a user request for data in order to get to the right datacenter in the cloud. The architecture can be subdivided into the following phases: data collection, data transfer, data processing, data categorization, big data storage and security (Thames and Schaefer 2017, p. 114–120).

3.2 Similar solutions for the problem in maintenance

After having discussed the state-of-the-art, a secure IoT architecture for remote maintenance that supports modern and legacy machine tools is presented. This technology can also be applied to upgrade older machine tools, which are not internet enabled, for instance, IoT devices. The resulting advantage is that older machine tools do not have to be exchanged but can be upgraded, which reduces costs. The architecture can be seen in Figure 21 and contains a broad overview with a dashboard for controlling, a cloud and its IoT units communicating securely via the internet or machine to machine protocols. Information from existing Enterprise Resource Planning (ERP) systems, Manufacturing Execution Systems (MES) or other systems

can be enriched with big data form cloud manufacturing services (Thames and Schaefer 2017, p. 236–238).

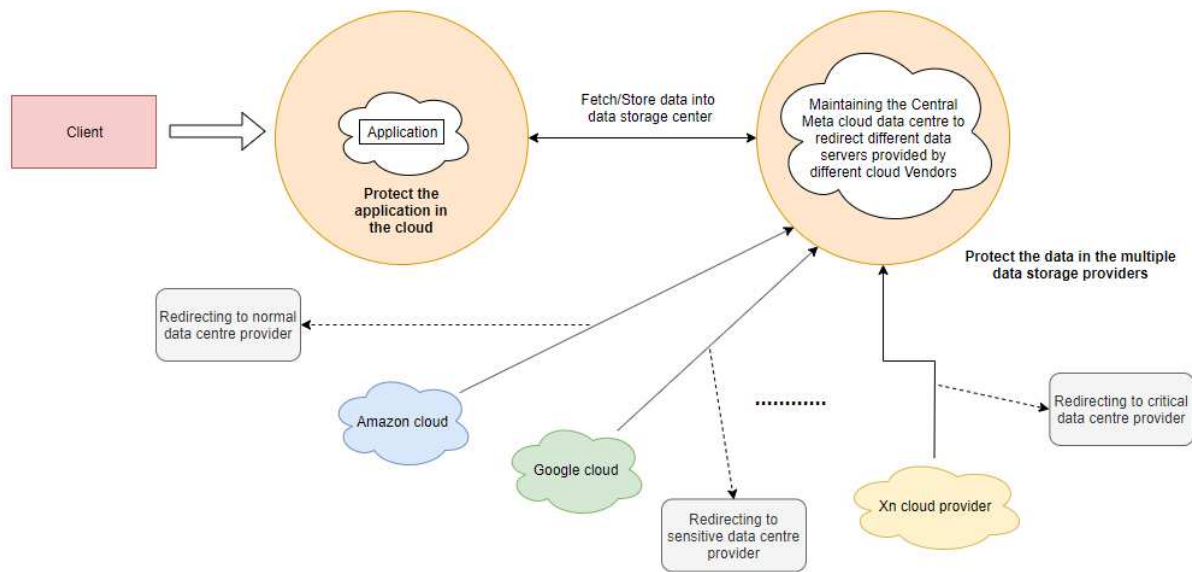


Figure 20 - Meta cloud data storage architecture, adapted from (Thames and Schaefer 2017, p. 115)

Remote maintenance can be understood as machine tool monitoring, data analysis with the help of thorough live service support and the actual maintenance of a machine tool. Live service support monitors the machines performance and predicts failure of machine tools or components. On a dashboard, these parameters are visualized and rule-based decision support can automate maintenance tasks.

With the architecture on Figure 21, a machine tool can be operated and maintained remotely, which saves costs, time as well as flexibility. Staff which used to do this maintenance task can now concentrate on areas where human thinking is needed and let the IoT devices do their work (Thames and Schaefer 2017, p. 236–238).

Concerning scalability, a modular approach is needed in order to be able to exchange and upgrade IoT devices fast. Figure 22 shows such an approach on a single IoT device about 100mm in size. This type of IoT unit can be used for sensing, actuating and communication on a M2M level or directly into the cloud. Standardized secure interfaces allow for a quick replacement of IoT units, which can also convert a communication protocol into another.

Data transfer works preferably wireless, but this approach allows for different types of communication like ZigBee, WiFi or others. In the middle of the IoT unit there is a data preprocessing unit that encrypts and decrypts data to ensure security. To be robust against physical attacks, the devices are small and only have a limited power supply and limited memory (Thames and Schaefer 2017, p. 236–240).

The next chapter will continue with other aspects relevant for this thesis focusing on practical blockchain solutions.

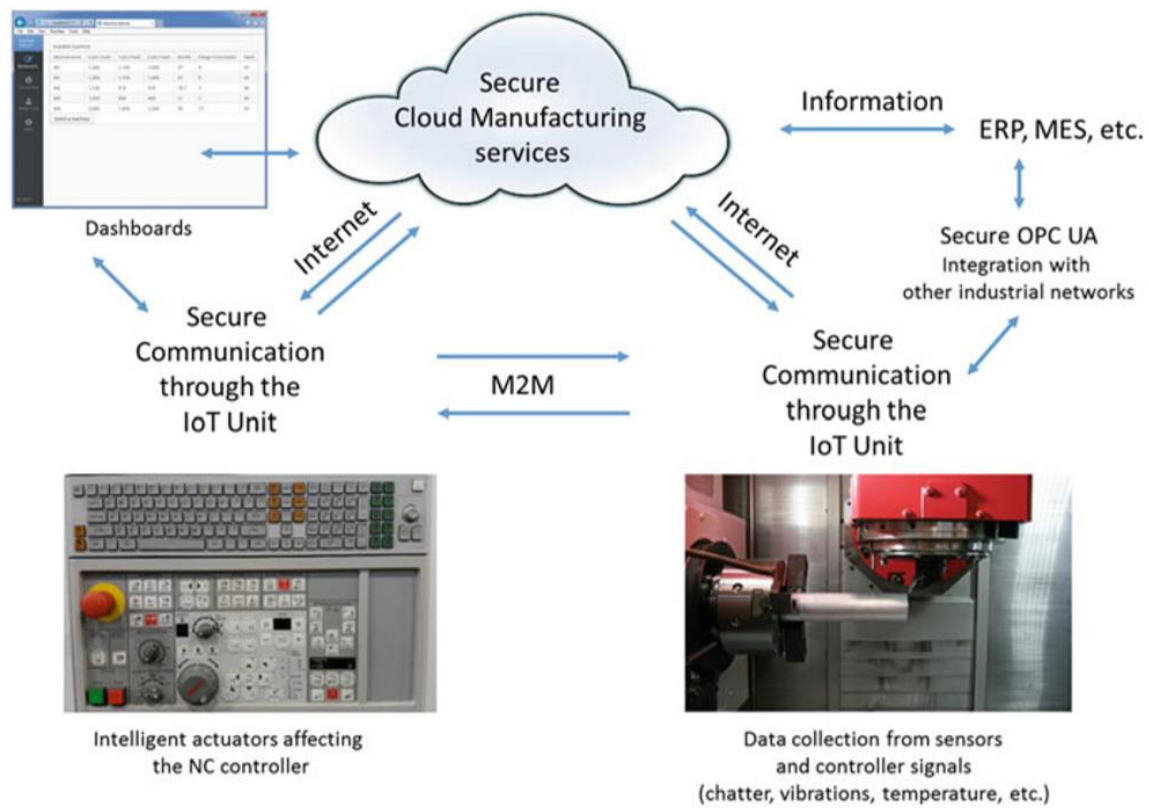


Figure 21 - IoT remote maintenance architecture (Thames and Schaefer 2017, p. 237)

3.3 Blockchain solutions in other sectors

In this chapter, other relevant work on the problem will be analyzed. It has to be mentioned, that not much research in blockchain technology applied to smart manufacturing exists, so following are solutions of other fields.

The public sector is one example in which blockchain technology is researched for the use of overcoming financial fraud. The Danish Tax Authorities have lost 1.8 billion USD by the end of August 2015 because of forged dividend tax refund applications. Fraudulent behavior is possible due to a problem with double spending, since there is no central information system that manages the data from all parties (Hyvärinen et al. 2017, p. 441–455).

Blockchain provides a perfect method of keeping track of transactions and can thereby be part of a solution for the double spending problem. It can even be assumed, that it is nearly impossible, since transactions are ordered chronological in the blockchain and must be verified by using a proof-of-work algorithm.

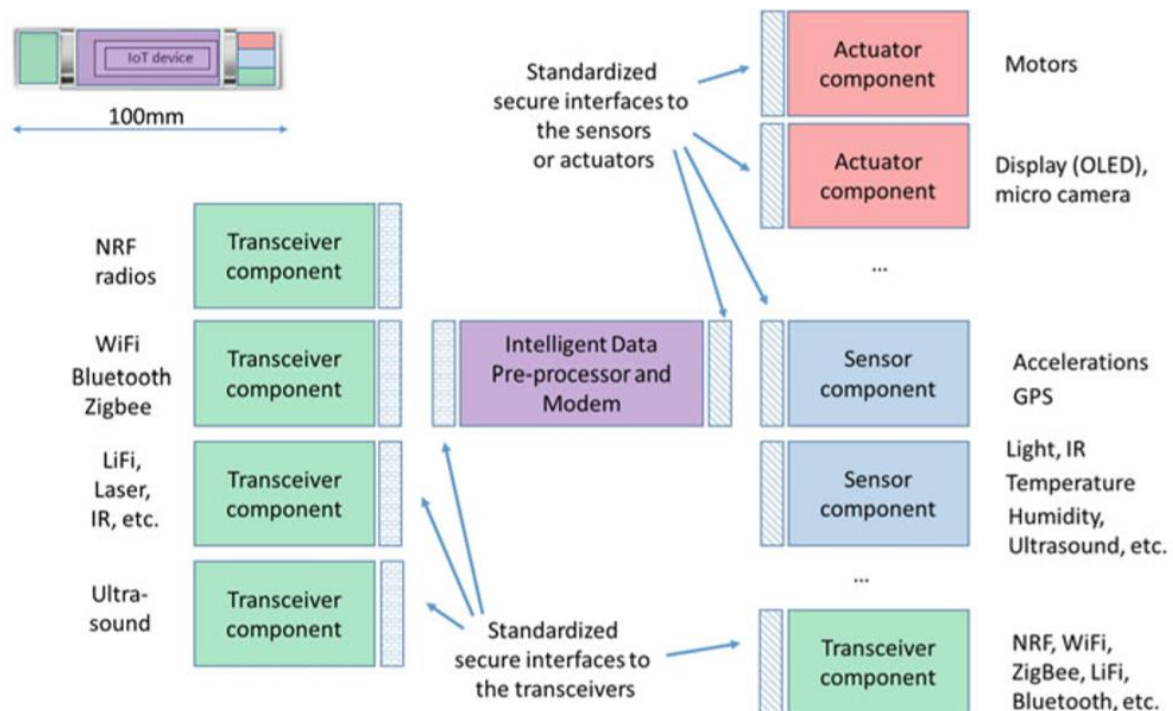


Figure 22 - Secure modular IoT Unit (Thames and Schaefer 2017, p. 238)

In the case of the Danish Tax Authorities, it would not be impossible to find every double spender but monitoring every transaction manually would be too much effort. Through the use of blockchain, it can be assured that double spending does not happen and there is a traceable history saved on the blockchain (Hyvärinen et al. 2017, p. 441–455).

Other important features of blockchain technology in this use-case are the fact that cryptographically linked transaction logs make manipulation unlikely, which leads to an immutability of those logs. Advantages such as decentralization also holds true in this use case, but end where the connection to the system of the Danish Tax Authorities occur.

The secure storage of this sensitive data, which can be provided by blockchain technology and facilitates data privacy, is highly critical. Due to the addition of smart contracts, no manual work is needed and a number of tasks can be completed automatically. All of this led to the implementation of a prototype to fight the double spending problem (Hyvärinen et al. 2017, p. 441–455).

The education sector and the internet could both benefit greatly from the introduction of blockchain. Qualifications from students could be stored on a blockchain and verified automatically, for instance, in case of a check-up for a job. Concerning the Internet, a public blockchain could solve the problem of censorship in countries where it is needed. In Table 24 potential application sectors for blockchain adoption and their possible impact can be found.

Table 24 - Potential application sectors and the impact

	Cost saving	Increased transparency	Increased security	Privacy
Smart manufacturing	X	X	X	X
Finance	X		X	
Public sector	X			
Intellectual property		X		
Education		X		
Internet		X		

4 Blockchain Collaboration Model

In this section, the proof of concept is provided by selecting a use case mentioned in section 0. The use case selected is the one related to maintenance.

In Figure 23, this use-case is presented, in which three companies use a blockchain for data sharing. Company X represents a smart factory in Austria that uses machines to bend steel bars. Company Y is an original equipment manufacturer in Italy that produces spare parts and Company Z is a company located in Great Britain that is responsible for maintenance.

Company X stores their machine data such as the status of the machine, the dimensions, color and others on the blockchain. Company Z does remote maintenance on the machines from Company X. They create tickets that store information regarding the procedure and can change the machine status if necessary. In this use case, there are the following statuses: Green (everything is fine), Yellow (minor issues) and Red (major issues).

In case Company Z needs spare parts for the maintenance procedure, they can choose one from a list of available products from Company Y. In case a product is not on the list, Company X can create a request for that spare part.

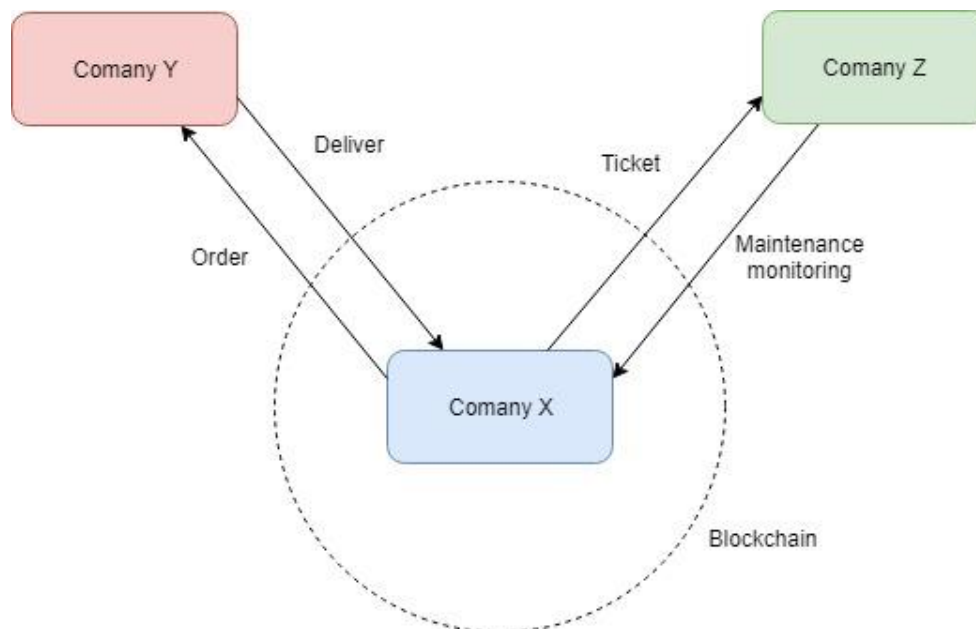


Figure 23 - Use-case blockchain network

Figure 24 shows the concept developed in this thesis in order to tackle the problem of data sharing between business partners distributed around the globe. Starting on the top there is a smart factory with machines using technology discussed in the theoretical part of this thesis such as CPS, Cloud, IoT and others. Different users from this factory access the blockchain with their mobile device or computer via a web application.

Every device represents a node in the blockchain network, which consists of smart contracts to handle the incoming data. Users can add machine data, spare part data, ticket data and request data. On the bottom is the original equipment manufacturer and the maintenance company, who also have access to the Ethereum blockchain.

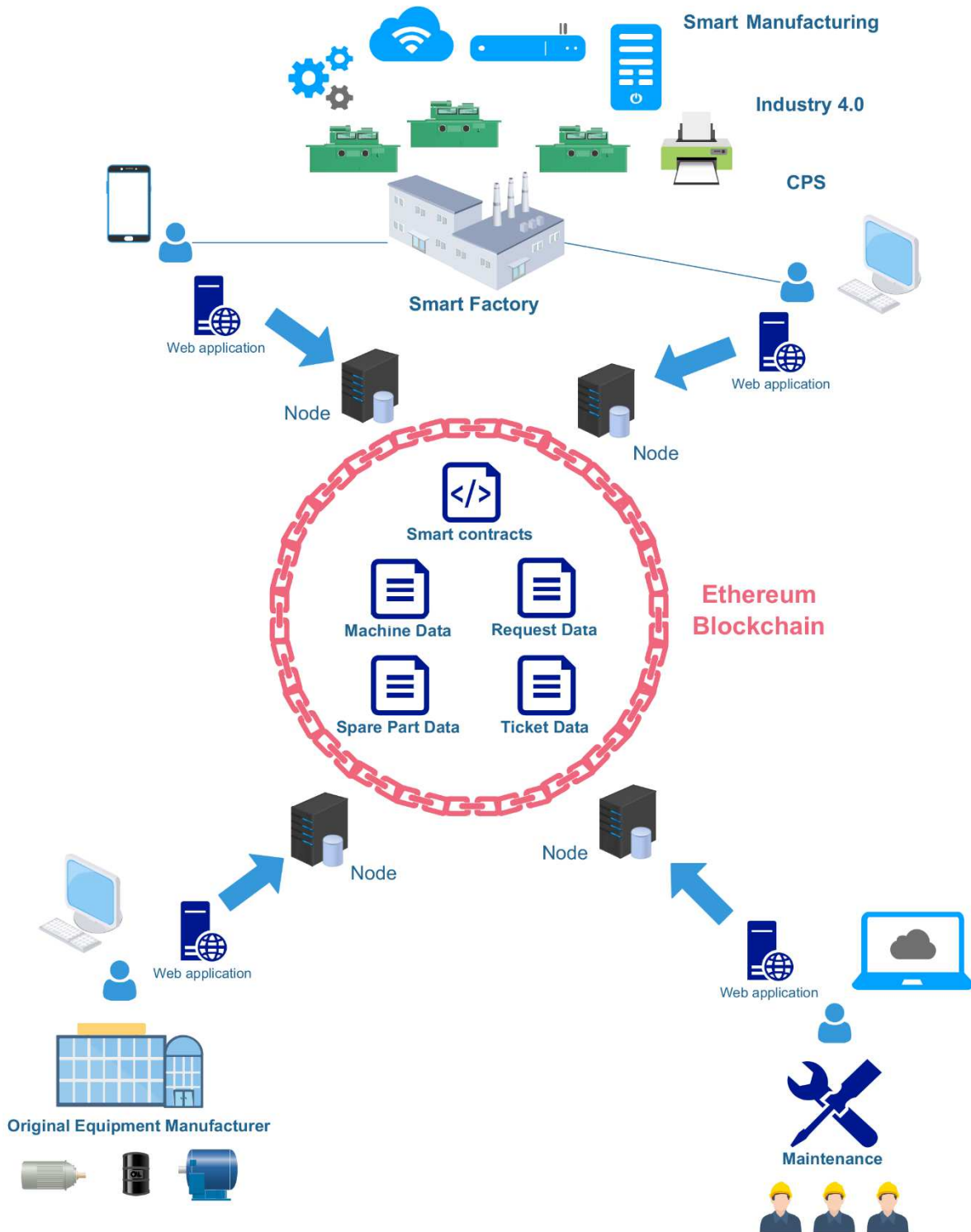


Figure 24 - Theoretical model

4.1 Concept and Specification

The following section will give a brief overview of the system, show different actions that can be performed and list non-functional requirements for the chosen use-case.

4.1.1 Blockchain System

The developed system consists of an Ethereum private blockchain, smart contracts and a front-end application. The blockchain was tested on a ganache test client and the smart contracts were compiled and deployed using Remix⁵.

The actual private blockchain would consist of multiple P2P nodes of different laptops or personal computers, which all installed the Ethereum client and have access to the blockchain. By becoming part of the same network, every user can now interact with the blockchain via smart contracts.

Smart contracts, as already explained in the theoretical part of this thesis, define the business logic. These contracts are deployed on the blockchain and since every user, who has access, has a full version of the blockchain. The deployment is not centralized as in traditional systems.

The programming language used for this blockchain is Solidity (Version 0.5.1). Solidity is a high level, object orientated language that enables the implementation of smart contracts. The language is similar to C++, JavaScript and Python.

A front-end application was developed for users to interact with the blockchain. Via this user-friendly GUI, a user can create machines, open details on them, create a ticket for a maintenance procedure and more. Meta Mask⁶ is needed for a transaction to actually work outside the test environment. Meta Mask can be installed as a Google Chrome addon and forms a bridge between Ethereum and web applications. After creating an account, the user can connect to a network. When starting a transaction, Meta Mask will pop up and a small amount of Ether has to be entered to start the verification process.

Figure 25 shows the Google Chrome add-on, which consists of the server to which it is connected in the top right corner, the account with the address and the amount of Ether this account has. The server chosen in this figure is the localhost on port 8545, which must match the server where the ganache client is running.

Ganache⁷ client is a blockchain emulator used for testing and the latest version of TestRPC. This emulator instantly mines transactions, has no transaction costs, allows

⁵ <https://remix.readthedocs.io/en/stable/>, remix.ethereum.org

⁶ <https://metamask.io/>

⁷ <https://docs.nethereum.com/en/latest/ethereum-and-clients/ganache-cli/>

to add Ether to accounts and allows for modifications of gas price as well as mining speed. For installation, the following command has to be entered into a console: “npm install -g ganache-cli”. The client then is started with the command: “ganache-cli”.

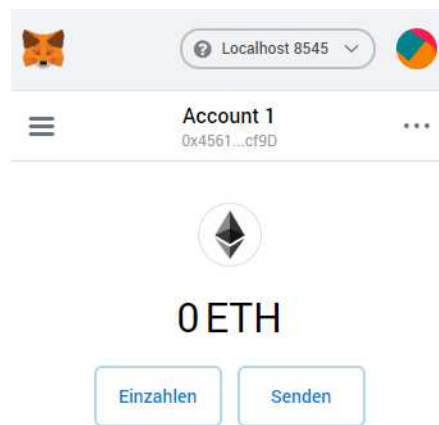


Figure 25 - Meta Mask Chrome add-on

The ganache client also creates ten accounts provided with 100 Ether and ten private keys. The starting output of ganache-cli can be seen in Figure 26.

```
C:\Users\glieb>ganache-cli
Ganache CLI v6.4.3 (ganache-core: 2.5.5)
net_version

Available Accounts
=====
(0) 0xd1031f953b13867d31b0efd099e5cd0fb5d0d1a4 (~100 ETH)
(1) 0x8c06168ffbe8ae935541e7a51091ad73a5a4862d (~100 ETH)
(2) 0xba276c591966c2e4e9782ae87fd7e8e95e15789f (~100 ETH)
(3) 0x6a4794bc3a470625b4826ae8c1f225061973cdd1 (~100 ETH)
(4) 0xf228d1ead65de88761d60987bda7acb7a029c5d6 (~100 ETH)
(5) 0x36df1a096ed52cf2b335660c8f72c8dcf92fbdc (~100 ETH)
(6) 0xe8f755ce1e7584248182f87585c165229cb1163b (~100 ETH)
(7) 0xbb913eedfb1ac1230e234b3dc9e81baf77d4e6bc (~100 ETH)
(8) 0xb786abee131f635827b3fd86cdcb011eedfb967 (~100 ETH)
(9) 0x0faefd0d6024e87e1be0acb20c67c9f63100f2cd (~100 ETH)

Private Keys
=====
(0) 0x6a1b81e49cf0052b44f6bba23eb85b9a82b9d6e4a79c9bccd80507f371b13819
(1) 0x818a2c5e6b92b68ee073bfc6481bb3b2dcfbeat90f722277aac1553396811b1f
(2) 0x2c67cfc74c9e580db6beeba562780d985b654dd78d0a22a198379e0aebc6ce81
(3) 0xa4ce941dac811d652c920dc903c17ba777b0ee1ac159bac9e84f014d3d3040f
(4) 0xbc20d1e03b9b31940f5c877efd47f87f1596bd158087318597c5f00b5b77372b
(5) 0xaea73571d9579b57d2b558e10d06417ffabd6596495a6ea617e280def5a2ba21
(6) 0x101221ac7bbbc18b5a46f036a3d5abe7f016bca4b2c5d62762c5db224b45024f
(7) 0x2751b748bb3b29258cb32b70a15e7b1d2ca8527d28a9e2dd3e99c378c2f959a3
(8) 0xe89f0102cdbac17ed1e2a3f51a6c6229ee34e288babf033033ce8f60a756e8be
(9) 0xd43c0fbe55a8d2c7e7545f7dfafb0027090380d23f84b01f61d8f462e7d5a534

HD Wallet
=====
Mnemonic:      race spirit cement area notable knife wink account chest mixed surprise galaxy
Base HD Path:  m/44'/60'/0'/0'/{account_index}

Gas Price
=====
20000000000

Gas Limit
=====
6721975
```

Figure 26 - Ganache client

Remix allows for writing Solidity contracts via a browser application in JavaScript. Additionally, Remix provides functionality such as debugging, testing and deploying of contracts. Figure 27 shows part of the Remix interface, where after the compilation of the contract for a machine, the contract can be deployed either on a random address or a specified one.



Figure 27 - Deploying a contract in Remix

4.1.2 Functional Requirements

A user can perform different actions on the blockchain by using the front-end application. An overview of each use-case can be seen in Figure 28 followed by a description of each case in the following tables.

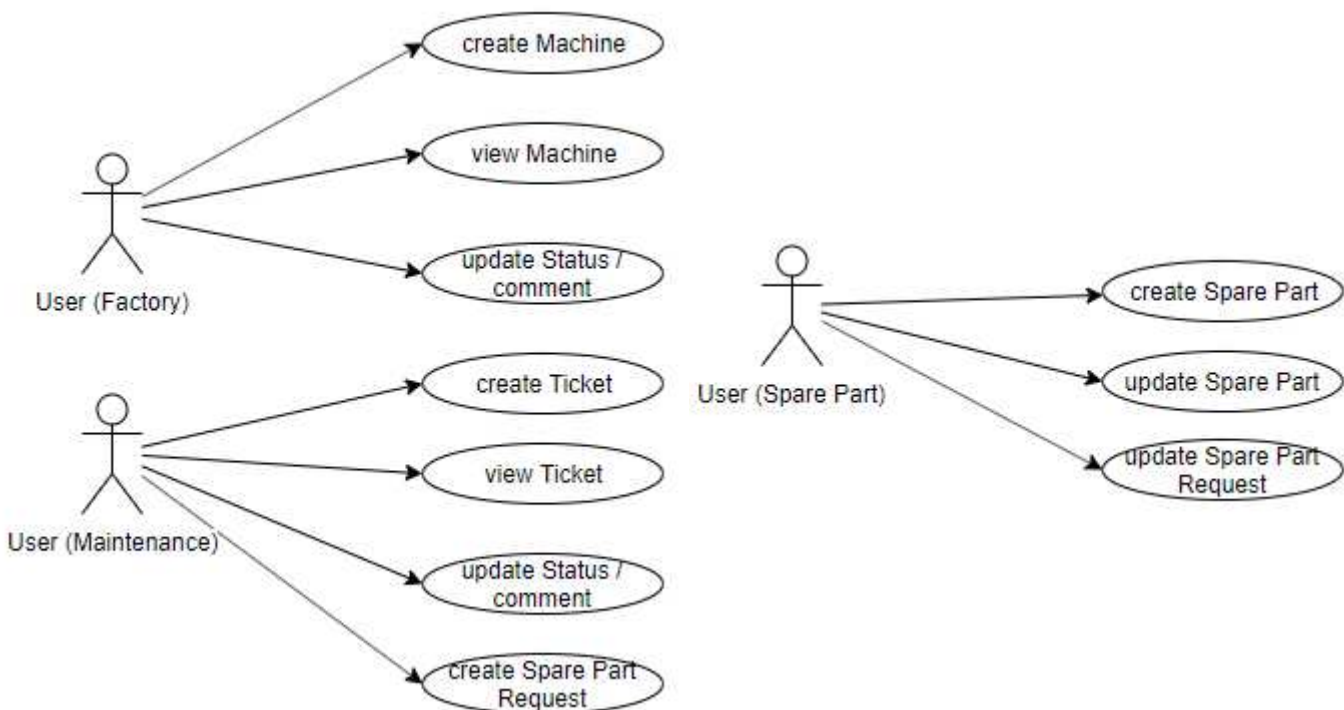


Figure 28 - User actions

To fulfil the functional requirements displayed in Figure 28, the following tables describe the main functions of the blockchain system developed in this thesis.

Starting off with Table 25 and Table 26, creating and viewing of a machine. A logged in user can create a machine, by entering the data necessary for the machine and saving it. This triggers a transaction on the blockchain, which is written into a block on the blockchain. After that this machine can be viewed, by calling a get method in the smart contracts.

Table 25 - Description of use-case Create Machine

Name	Create Machine
Goal	Creating a machine
Actor	User
Pre-Condition	User must be logged in
Post-Condition	Machine with variables created
Normal case	<ul style="list-style-type: none"> • User fills in fields in GUI and presses the button save • Transaction is sent • Transaction is mined • Result is the transaction hash

Table 26 - Description of use-case View Machine

Name	View Machine
Goal	Viewing a machine
Actor	User
Pre-Condition	User must be logged in
Post-Condition	Machine with variables displayed
Normal case	<ul style="list-style-type: none"> • User clicks on a machine in GUI he wants to view • Transaction is sent • Transaction is mined • Result is the transaction hash & the values of the machine

In Table 27 the function of updating the status and adding of a comment is described. This functionality only works in combination with the creation of a Ticket of a machine. With the adding of such a ticket, the status and the comment of a machine can be adapted. For instance, the machine status can be changed from green to red, which means, that the machine has functional issues.

The functionality of adding and viewing a Ticket is depicted in Table 28 and Table 29. A Ticket can only be added to an already existing Machine. Doing so, a maintenance procedure can be documented, by adding a Ticket to a Machine. There, data like the type of procedure, a requested Spare Part and other information can be added. By saving the Ticket a transaction on the blockchain is triggered again. After that, the created Ticket can be viewed, by clicking on the Ticket.

Table 27 - Description of use-case Update status or comment

Name	Update status or comment
Goal	Updating the status or comment
Actor	User
Pre-Condition	User must be logged in, Machine is already created
Post-Condition	Status or comment updated, new Ticket was created
Normal case	<ul style="list-style-type: none"> • User creates a new ticket for a machine in GUI, fills in the status or comment and presses the button save • Transaction is sent • Transaction is mined • Result is the transaction hash

Table 28 - Description of use-case Create Ticket

Name	Create Ticket
Goal	Creating a ticket
Actor	User
Pre-Condition	User must be logged in
Post-Condition	Ticket with variables created
Normal case	<ul style="list-style-type: none"> • User fills in fields in GUI and presses the button save • Transaction is sent • Transaction is mined • Result is the transaction hash

Table 29 - Description of use-case View Ticket

Name	View Ticket
Goal	Viewing a ticket
Actor	User
Pre-Condition	User must be logged in
Post-Condition	Ticket with variables displayed
Normal case	<ul style="list-style-type: none"> • User clicks on a ticket in GUI he wants to view • Transaction is sent • Transaction is mined • Result is the transaction hash & the values of the ticket

While creating a Ticket, a Spare Part can be requested. This functionality can be found in Table 30. This would mean, that a procedure needs a Spare Part, that isn't on stock and is requested. For that an entry into the blockchain is entered.

Table 30 - Description of use-case Create Spare Part Request

Name	Create Spare Part Request
Goal	Creating a Spare Part Request
Actor	User
Pre-Condition	User must be logged in
Post-Condition	Spare Part Request with variables created
Normal case	<ul style="list-style-type: none"> • User fills in fields in GUI and presses the button save • Transaction is sent • Transaction is mined • Result is the transaction hash

Table 31 and Table 32 show the creation and updating of a Spare Part. A new Spare Part can be created, by adding a name and a description, as well as an amount, which is on stock. By adding the amount of zero, the Spare Part would be unavailable. When a number greater than zero is entered, the Spare Part would be available. It is also possible to restock a Spare Part, by clicking on it and entering the amount that should be added to the stock.

Table 31 - Description of use-case Create Spare Part

Name	Create Spare Part
Goal	Creating a Spare Part
Actor	User
Pre-Condition	User must be logged in
Post-Condition	Spare Part with variables created
Normal case	<ul style="list-style-type: none"> • User fills in fields in GUI and presses the button save • Transaction is sent • Transaction is mined • Result is the transaction hash

Table 32 - Description of use-case Update Spare Part

Name	Update Spare Part
Goal	Updating the Spare Part
Actor	User
Pre-Condition	User must be logged in
Post-Condition	Spare Part updated
Normal case	<ul style="list-style-type: none"> • User clicks on Spare part he wants to update, enters the amount to restock the Spare Part and saves • Transaction is sent • Transaction is mined • Result is the transaction hash

Lastly, Table 33 shows the updating of a Spare Part Request. Fulfilling a request with a Spare Part, which is in stock, with a certain amount, will close that Spare Part Request and add the Spare Part to the Ticket, in which it was requested.

Table 33 - Description of use-case Update Spare Part Request

Name	Update Spare Part Request
Goal	Updating the Spare Part Request
Actor	User
Pre-Condition	User must be logged in
Post-Condition	Spare Part Request updated
Normal case	<ul style="list-style-type: none"> • User clicks on Spare part Request he wants to update, enters the Spare Part, enters the amount and saves • Transaction is sent • Transaction is mined • Result is the transaction hash

4.1.3 Non-Functional Requirements

Along with the functionality described in the previous chapter, there are some non-functional requirements. Thinking about the challenges of the adaption of blockchain, usability, data privacy and data immutability are one of the most important non-functional requirements. Usability is key to reduce the fright of using a technology and data privacy and immutability are important to fulfil security requirements. These and other non-functional requirements can be found in Table 34.

Table 34 - Non-functional requirements

Non-Functional Requirements	Description
Usability	Usability is an important factor when it comes to blockchain, since the concept itself is new and not many people are familiar with it. A well-designed user interface can make it, so that the user only comes into contact with the blockchain when they have to use Meta Mask.
Data integrity	The data integrity of on-chain data must be ensured.
Data privacy	Only members of the blockchain should be able to read data stored on the blockchain.
Availability	The blockchain must be available in terms of responsiveness to enable read and write operations.
Transparency	Every member of the private blockchain should be able to comprehend changes to data on the blockchain.

Non-Functional Requirements	Description
Data immutability	Data should not be changed after its creation, so no untraced changes can occur and security is increased. This also enhances trust between the participants of the blockchain because no one can delete or change data with bad intentions.

4.2 Design

This chapter focuses on the system architecture and the components involved as well as the data model used for the implementation of the prototype.

4.2.1 Architecture

The architecture used for the prototype can be seen in Figure 29 and consists of a presentation layer, an account management layer and a blockchain layer.

The presentation layer consists of the html files for the GUI, a CSS file, which handles the style, Web3.js, which is the connection to the blockchain and the contract ABI. More information to the front-end part of the prototype will be given in chapter 4.3.3.

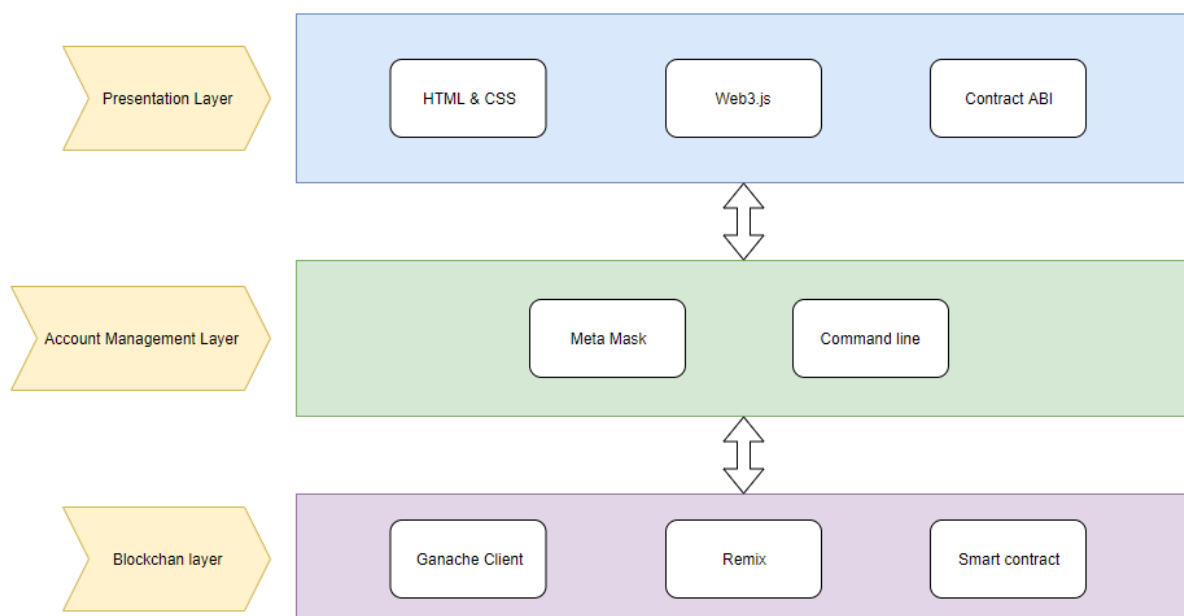


Figure 29 - System architecture

The account management layer consists of Meta Mask, the basic command line as well as ganache. Meta Mask is needed for confirming transaction and can host different accounts with an amount of Ether. Ganache creates ten accounts, which makes it part

of the account management layer. Also, the command line can be used to create accounts.

Finally, the blockchain layer consists of the ganache client, Remix and smart contracts. The ganache client represents the blockchain and sets gas limit and price. Remix is used to connect to ganache as well as to compile and deploy smart contracts.

4.2.2 Data Model

The data model of the program can be seen in Figure 30. It contains machine, ticket, spare part and spare part requests. In the machine object, all data connected to the machine is saved such as the dimension, weight of the machine, tolerances and others. Additionally, a status, a comment and a list of tickets is stored.

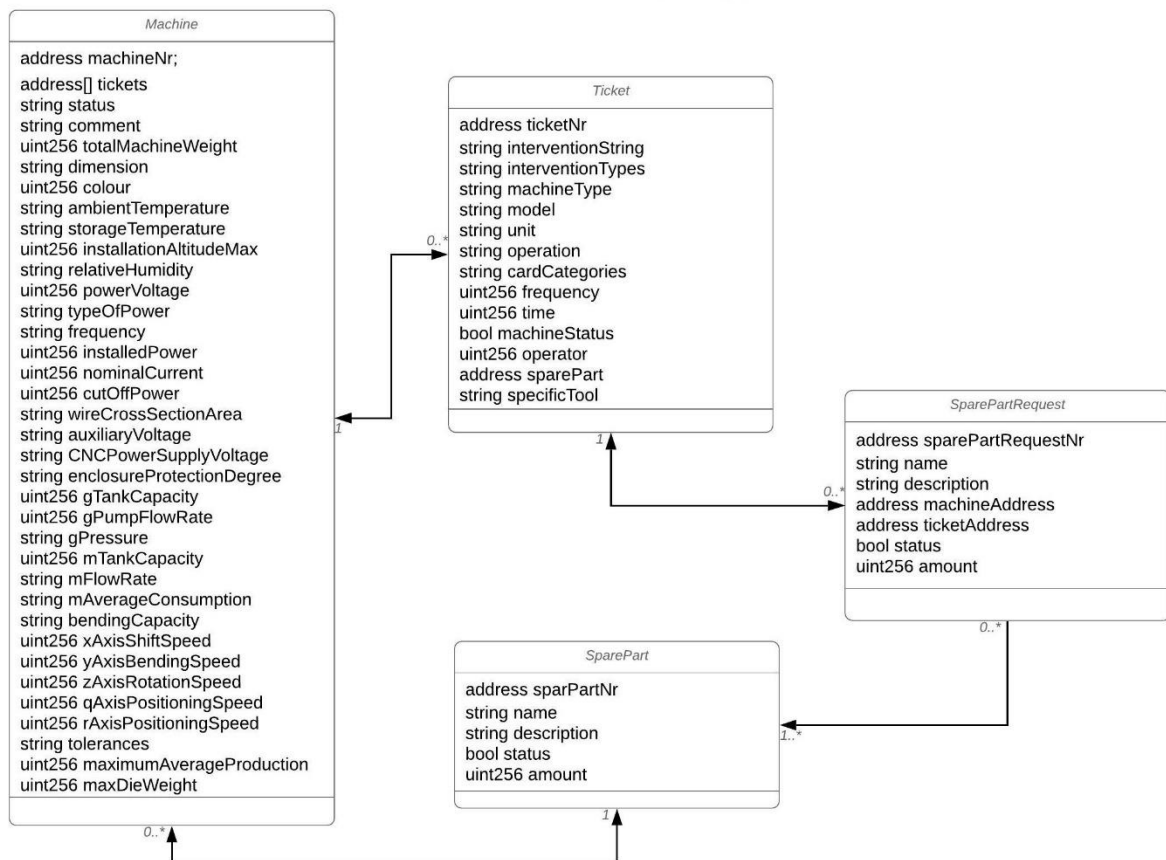


Figure 30 - Data model

Tickets can be created for a machine, to document maintenance procedures. Therefore, a ticket contains information such as the intervention type, the frequency of the procedure, the time it took and others. Also, a ticket can contain a required spare part, which can either be on stock or has to be requested.

Spare parts contain information such as the name, a description and an amount available. If there is no spare part in store, it can be requested. A request contains the name, a description, a status (if the request is resolved or not) and an amount.

In all four objects, addresses are saved. MachineNr, ticketNr, sparePartRequestNr and sparePartNr can be seen as IDs and pointers to where the data is saved. So, every machine that is created, gets a random unique address. A machine also saves all tickets associated to it, or more precisely the addresses of all tickets. Lastly, a spare part request also saves the address of the machine and ticket it was created for.

In this data model the following operations can be performed:

- Machines can be created through different parameters
- Tickets for machines can be created through different parameters
- Spare parts can be created through different parameters
- Spare part requests can be created through different parameters and fulfilled
- Data from all objects can be updated

This data model provides the basic structure for the chosen use-case and lays the foundation for the implementation explained in the next chapter.

4.3 Implementation

In this chapter, the implementation of the prototype will be shown with some examples of smart contracts and the front-end application. The implementation is divided into three phases being: Configuration, Smart contract and front-end.

4.3.1 Configuration

The configurations of the private Ethereum blockchain are as follows:

1. Node.js needs to be installed
2. Npm needs to be installed
3. Run `npm install -g ethereumjs-testrpc`
4. Run Ganache-cli to start the local server
5. Make sure to set the environment in Remix to Web3 Provider in case of using the test setup or Injected Web3 to use a server over Meta Mask

The full guide on how to start the prototype can be found in the attachments. With this configuration, the programming of the smart contracts and the front-end can begin.

4.3.2 Smart Contracts

Since the blockchain was programmed on Ethereum, the language used for the smart contracts is Solidity. In Listing 1 the smart contract for spare parts can be seen. In the first line, version 0.5.1 can be found.

The object SparePart is defined in line 4-10 with its variables sparePartNr, name, description, status and amount. Next, in line 11 a mapping is defined, which is a list of spare parts in this case and an array containing addresses of spare parts is defined in Line 12.

Following are functions to set and get the data of the object. These functions always need an address to either find the right object or to save a new object with a specified address. Lastly, there is a function to count the amount of spare parts stored as well as a function that returns all addresses of spare parts.

Interestingly, Solidity cannot have too many input parameters or the code will not compile. For this reason, the functions for setting and getting data are split up. Other contracts can be found in the attachments, but they follow a similar fashion.

Listing 1 - Smart contract for Spare parts

```

1.  pragma solidity ^0.5.1;
2.
3.  contract SparePartContract {
4.      struct SparePart {
5.          address sparPartNr;
6.          string name;
7.          string description;
8.          bool status;
9.          uint256 amount;
10.     }
11.     mapping (address => SparePart) sparePartList;
12.     address[] sparePartData;
13.
14.     function setSparePartData1(address _sparePartNr,
15.                               string memory _name,
16.                               string memory _description,
17.                               bool _status,
18.                               uint256 _amount
19.                               ) public {
20.
21.         SparePart storage sparePart = sparePartList[_sparePartNr];
22.         sparePart.name = _name;
23.         sparePart.description = _description;
24.         sparePart.status = _status;
25.         sparePart.amount = _amount;
26.
27.         sparePartData.push(_sparePartNr) -1;
28.     }
29.     function getSparePartData1(address _sparePartNr) public view returns (
30.                                     string memory _name,
31.                                     string memory _description,
32.                                     bool _status,
33.                                     uint256 _amount){
34.         return (sparePartList[_sparePartNr].name,
35.                 sparePartList[_sparePartNr].description, sparePartList[_sparePartNr].status,
36.                 sparePartList[_sparePartNr].amount);

```

```

37.     }
38.     function setStatus(address _sparePartNr,
39.                       bool _status
40.                       ) public {
41.         SparePart storage sparePart = sparePartList[_sparePartNr];
42.         sparePart.status = _status;
43.     }
44.     function setAmount(address _sparePartNr,
45.                       uint256 _amount
46.                       ) public {
47.         SparePart storage sparePart = sparePartList[_sparePartNr];
48.         sparePart.amount = _amount;
49.     }
50.     function getAmount(address _sparePartNr) public view returns (uint256 _amount){
51.         return (sparePartList[_sparePartNr].amount);
52.     }
53.     function countSpareparts() view public returns (uint) {
54.         return sparePartData.length;
55.     }
56.     function getSparepartAddress() view public returns (address[] memory) {
57.         return sparePartData;
58.     }
59. }

```

4.3.3 Front-End

For the front-end basic html with JQuery and web3js were used. JQuery is a JavaScript library adding functionality to the HTML sites. Web3js is an Ethereum JavaScript API which has a collection of libraries that enable the interaction with Ethereum nodes using mainly HTTP. The Web3 providers can be set in the following way shown in Listing 2.

Listing 2 - Section of front-end code

```

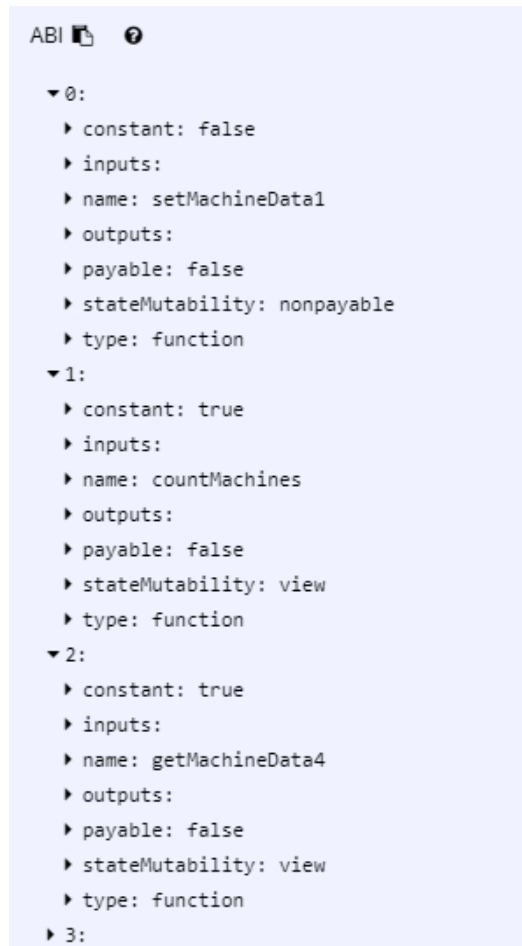
1. if (typeof web3 !== 'undefined') {
2.     web3 = new Web3(web3.currentProvider);
3. } else {
4.     // set the provider you want from Web3.providers
5.     web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));
6. }
7. web3 = web3.eth.defaultAccount = web3.eth.accounts[0];
8.
9. var TicketContract = web3.eth.contract(...ABI...);
10. var MachineContract = web3.eth.contract(...ABI...);
11. var SparePartContract = web3.eth.contract(...ABI...);
12. var SparePartRequestContract = web3.eth.contract(...ABI...);
13.
14. var Ticket = TicketContract.at('0x7e06ccb5c46db5ed809d9d8f73132608495f894c');
15. var Machine = MachineContract.at('0x7d51587098e47cbc2b3467e2e1eb3ab4a49650ad');
16. var SparePart = SparePartContract.at('0x82e4fb55e67dd09caa29ea1a7dc283180c136396');
17. var SparePartRequest = SparePartRequestContract.at('0x27c51c3ba389002c63b2715dfc7587aa04eb459c');
18.
19. var ticketAddresses = Ticket.getTicketAddress();
20. var machineAddresses = Machine.getMachineAddress();

```

In line 9-12 of Listing 2, the contract ABIs are defined. Instead of ...ABI... the actual application binary interface of the smart contracts is used. For Ethereum, the ABI is used to encode Solidity contract calls and to receive data stored on the blockchain.

This thesis uses the platform Remix. On the platform the ABI of a contract can be extracted after compiling a contract. This can be seen in Figure 31.

Line 14-17 of Listing 2 are the addresses where the contract is deployed. After successfully compiling the contract, Remix can deploy a contract either on a random address or a specified one.



```

ABI
  0:
    constant: false
    inputs:
    name: setMachineData1
    outputs:
    payable: false
    stateMutability: nonpayable
    type: function
  1:
    constant: true
    inputs:
    name: countMachines
    outputs:
    payable: false
    stateMutability: view
    type: function
  2:
    constant: true
    inputs:
    name: getMachineData4
    outputs:
    payable: false
    stateMutability: view
    type: function
  3:
  
```

Figure 31 - Part of the ABI from the Machine contract in Remix

Listing 3 shows the part of the code with which a Spare Part is created via the GUI. In line 2-9 the input labels are defined as well as the save button. Line 11-25 represents the script so save data in the blockchain. At first, a random address is created, then the values of the input field are used to call the function setting data for the Spare Part.

Listing 3 - Section of HTML file for creating a Spare Part

```

1. <body>
2.   <label for="name" class="col-lg-2 control-label">name</label>
3.   <input id="name" type="text">
4.   <label for="description" class="col-lg-2 control-label">description</label>
5.   <input id="description" type="text">
6.   <label for="amount" class="col-lg-2 control-label">amount</label>
7.   <input id="amount" type="number">
8.
9.   <button id="button">Save</button>
10.

```

```

11.   <script>
12.     $("#button").click(function () {
13. var   rand = "0x1111111111111111111111111111111111111111111111111111111111111111".replace(/1/g, function () {
    return (~(Math.random() * 16)).toString(16); });
14.     var   addr = web3.personal.newAccount(rand);
15.     var   status;
16.     if ($("#amount").val() > 0) {
17.         status = true;
18.     }
19.     else {
20.         status = false;
21.     }
22.
23. SparePart.setSparePartData1(addr, ($("#name").val()), ($("#description").val()), status,
    ($("#amount").val()))
24. }
25. </script>
26. </body>

```

Listing 4 shows how a Spare Part can be viewed. The call “getSparePartData1” sends a request to receive the Spare Part data from the specified address. The data is structured in an array and is extracted in line 10-13.

Listing 4 - Section of HTML file for viewing a Spare part

```

1. var sparePartData1 = SparePart.getSparePartData1(sparePartAddress);
2. var status;
3.
4. if( sparePartData1[2] == true){
5.     status = "Available";
6. }
7. else{
8.     status = "Unavailable";
9. }
10. $("#sparePart").html('Name: ' + sparePartData1[0] +
11.                        '<br> Description: ' + sparePartData1[1] +
12.                        '<br> Status: ' + status +
13.                        '<br> Amount: ' + sparePartData1[3]
14.                        );

```

After finishing these steps, the smart contract, which is deployed on the blockchain, is connected with the HTML file.

The front-end for Company X and Z can be seen in Figure 32. Three different machines can be seen with a status and a comment. On each machine, it is possible to add a ticket by clicking on the “New Ticket” button. The stored tickets of a machine are listed below each machine.

On the bottom of the dashboard, open requests for Spare Parts are provided. By clicking on a Machine (Figure 33), Ticket (Figure 34) or Spare Part Request additional details are shown.

In Figure 33, the details of a Machine structured in physical characteristics, working environment, electrical system, grease lubrication system, mineral lubrication system and working characteristics can be seen.

Machine dashboard

The dashboard displays the following information:

- Machine Nr: 1**
 - Status: **Green**
 - Comment: everything works
 - New Ticket
 - Ticket Nr: 1
 - Ticket Nr: 2
- Machine Nr: 2**
 - Status: **Yellow**
 - Comment: Small problems with light curtains
 - New Ticket
 - Ticket Nr: 1
 - Ticket Nr: 2
 - Ticket Nr: 3
 - Ticket Nr: 4
- Machine Nr: 3**
 - Status: **Red**
 - Comment: Failure imminent
 - New Ticket

Buttons: New Machine

Open Requests

- Spare Part Request Nr: 1

Figure 32 - Machine dashboard

In Figure 34, the details of a Ticket are listed. In this case, Company Z carried out preventive maintenance on a tube bender model Ele101 on the Axes x and e. The machine kept running during the procedure, a Spare Part was requested from Company Y and a specific tool was used for the operation. Additionally, it is possible to download the Ticket as a pdf by pressing the button “Save as PDF” seen on the bottom left of Figure 34.

Figure 35 shows Company Y’s view, which is responsible for the Spare Parts of the machines. Eight different Spare Parts with a name and a status are displayed. If the Spare Part is in stock, it is available. Otherwise, it is unavailable and needs to be restocked. This can be done by clicking on the Spare Part, which opens the view of Figure 36, where a restock option is available.

Additionally, Figure 35 shows Spare Part Requests, which can be open, or closed. On the top right side, a pie chart visualizes the open and closed requests. Open requests for Spare Parts can be fulfilled, by clicking on the button “Fulfil Request”, which opens the view of Figure 38. There, a Spare Part can be chosen out of a list and the requested amount can be entered. In case there are not enough spare parts, a warning is shown.

Machine 1

Back

Status: Green
 Comment: everything works

Physical characteristics

Total machine weight (daN): 4700
 Dimension (l x d x h / mm): 4400 x 2250 x 1900
 Colour (RAL): 9017

Working environment

Ambient temperature (°C): +5 / +40
 Storage temperature (°C): -25 / +55 (Max. 70°C for 24h)
 Installation altitude (m): 1000
 Relative humidity (%): 5-75 (max. for 24h)

Electrical system

Power voltage (V): 400
 Type of power: Three-phase without neutral
 Frequency (Hz): 50 +/- 0.1%
 Installed power (kVA): 22
 Nominal current (A): 35
 Cut-off power (kA): 35
 Wire cross-section area (mm²): Cf. wiring diagram
 Auxiliary voltage (V): -
 CNC power supply voltage (V): 24 dc
 Enclosure protection degree (A): IP54

Grease lubrication system

Tank capacity (dm³): 2
 Pump flow rate (cm³/min): 2
 Pressure (bar): 20 max.

Mineral lubrication system mandrel

Tank capacity (l): 0
 Flow rate (l/min): see settings
 Average consumption (l/min): see settings

Working characteristics

Bending capacity (Nm): 4750
 X axis shifting speed (mm/sec): 2000
 Y axis A/R bending speed (°/sec): 150
 Z axis rotation speed (°/sec): 630
 Q axis positioning speed (mm/sec): 1300
 R axis positioning speed (°/sec): 115
 Tolerances (mm): +/-0.05
 Maximum average production (bends/h): 1800
 Maximum die weight (kg): 35

Figure 33 - Machine detail view

Ticket 1

Card title:	Preventive maintenance
Intervention Types:	Lubrication, Inspection
Machine Type:	Tube bender
Model:	Ele101
Unit:	Axes x,e
Operation description:	Grease cartridge replacement
Card Categories:	Quality
Frequency:	6
Time:	10
Machine status:	Running
Number of operator:	1
Spare part:	Request New- Grafloscon AG2 Ultra cartridge
Specific tool:	Standard mechanical equipment

Save as PDF Back

Figure 34 - Ticket detail view

Spare Parts

- **Spare Part Nr: 1**
Name: Grafloscon AG2 Ultra cartridge
Status: Unavailable
- **Spare Part Nr: 2**
Name: GRAFLOSCON SY 20 ULTRA
Status: Available
- **Spare Part Nr: 3**
Name: AMBLYGON TA 15/2
Status: Available
- **Spare Part Nr: 4**
Name: FLUOROPAN T 20
Status: Unavailable
- **Spare Part Nr: 5**
Name: LAMORA D 100
Status: Available
- **Spare Part Nr: 6**
Name: OPTAPLUS AO 35
Status: Available
- **Spare Part Nr: 7**
Name: Redaelli 9 R 100
Status: Available
- **Spare Part Nr: 8**
Name: STABURAGS N 12 MF
Status: Available

[New Spare Part](#)

Requests for Spare parts

- **Spare Part Request Nr: 1**
Status: Open
[Fulfill Request](#)
- **Spare Part Request Nr: 2**
Status: Closed

Requests

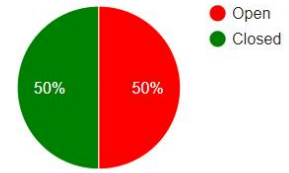


Figure 35 - Spare Part dashboard

Spare Part 1

Name: Grafloscon AG2 Ultra cartridge
Description: or equivalent
Status: Unavailable
Amount: 0

Restock

Amount of Spare parts

 [Save](#)

[Back](#)

Figure 36 - Spare Part detail

Spare Part Request

Name

Description

Amount

Save Spare Part Request

Figure 37 - Spare Part Request

Spare Part

Amount

Save Back

Figure 38 - Fulfil Request

In the next chapter, the validation of this prototype is described by comparing state of the art solutions with this prototype.

4.3.4 Evaluation and Validation

In this section, the proposed architecture will be evaluated in terms of the non-functional requirements as well as the functionality required for the use-case and a qualitative analysis. The evaluation is based on the proof-of-concept.

The functional requirements from section 4.1.2 and their fulfilment are as follows:

- Writing data on the blockchain is possible due to the “setter” functions of the smart contracts.
- Reading data on the blockchain is possible due to the “getter” functions of the smart contracts. Additionally, every object has a unique ID for referencing and a container, in which it is saved.

- Updating data on the blockchain is also possible due to the set functions of the smart contracts. However, it has to be mentioned, that old data can always be found on the blockchain, since the data is not actually replaced, rather a new transaction is created.

The non-functional requirements from section 4.1.3 and their fulfilment are as follows:

- Usability is granted, because users only encounter the blockchain, when they want to use Meta Mask for verifying a transaction. Everything else is provided via a basic html webpage. Further evaluation can be found in the alpha and beta test in this chapter.
- Data integrity is provided as long as no user of the blockchain makes wrong changes to data. If this occurs, the correct data can still be apprehended due to the data immutability of blockchain data.
- Data Privacy is provided due to the fact that a private blockchain is used, which means data is not available to the public. In this case data can only be shared among participants of the blockchain, which are users from the three companies.
- To ensure availability, sufficient replication is needed.
- Transparency is provided by the characteristics of blockchain technology. Once a transaction is mined and permanently saved into a block, each member of the blockchain gains access to it. Therefore, all changes can be traced back if needed.
- Data immutability is also provided by the nature of blockchain, since a transaction cannot be changed once it is saved in a block.

After evaluating the functional and non-functional requirements, a qualitative analysis of the proof-of-concept was carried out. A use case from the state of the art was compared to the proof-of-concept. In section 3.1.1, a design for a collaboration scenario was explained, in which a model owner creates a CAD model and encrypts as well as uploads it to a cloud. A collaborator then can decrypt the file and access this data.

The main requirements for ensuring security during data sharing are the following two (Rutledge and Hoffman 1986, p. 296–307):

- Information hiding: an unauthorized user must not be able to access confidential information
- Information authentication: information has a verification capacity which ensures that the information has not been altered

The CAD model design from the state of the art achieved this through a watermark, access control, a multi-resolution approach and encryption of data. The watermark

ensures data integrity and protects intellectual property. Access is controlled by a user's authorization. The multi-resolution approach enables collaborative work on a model and data encryption is used for data hiding.

The proof-of-concept proposed in this thesis also ensures information hiding and information authentication through the concept of blockchain. Since a private blockchain is used, information can only be accessed by members of the blockchain. Information authentication is granted because information on the blockchain cannot be altered and every change can be reconstructed. Time stamps and signing of transactions enable transparency.

In conclusion, both concepts can be used to tackle the problem of information sharing, but blockchain has all the mechanisms built in the roots of its technology. For the CAD design, additional software is required to ensure safety during collaboration on a project.

For the validation of the software functionality the IEEE standard for software verification and validation was used (IEEE 2016). A functional as well as a usability test was performed by firstly doing an alpha and then a beta test.

The alpha test was conducted with a research assistant of the TU Wien. For this, a laptop was used, where only the operation system Windows 10 was installed. For the installation of the prototype, the provided installation guide, which can also be found in the attachments, was used. The installation steps were followed, but there were some problems with the compilation of the smart contracts with remix. The difficulties were solved by using Google Chrome instead of Mozilla Firefox. After that the prototype was running on the laptop. The installation process took about an hour, including the installation of basic software like Google Chrome.

Next was the testing of the functionality. The main tasks like creating a new Machine, adding a Ticket with a Spare Part Request and the adding of a Spare Part were performed successfully. The feedback was mainly regarding the installation process, since it took longer than expected. So, the installation guide was reworked to foresee possible problems in the process.

For the beta test five participants were given the reworked installation guide, a questionnaire and the needed files to run the prototype. The background of these people ranged from a very low IT affinity to a very high IT affinity, in this case a software developer. The task for these people were to firstly to set up the prototype, test the functionality and lastly rate the usability.

The metrics used for the beta test were:

- Successful task completion

- Errors
- Error-free rate
- Time on task
- Subjective measures

Starting off with the installation process, participants took from 20 minutes to 38 minutes. The participants with lower IT affinity had a harder time with the installation and needed some basic help like starting the command line. Other than that, the developer had some issues with the installation, since he had an old version of NPM already installed: He ran into problems while trying to install the ganache client. Only after updating NodeJS, NPM and deleting the old installation of ganache, was he able to get the prototype running. Comparing the times of installation with the IT affinity level, the people with a higher level were faster overall.

Following was the functional test, where nine test scenarios were listed with the exact steps and expected results. The participants had to check if each step was successful. The test scenarios were:

- Add a new Machine
- View a created Machine
- Add a new Ticket
- View a created Ticket
- Create a Spare Part Request
- Add a new Spare Part
- View a created Spare Part
- View and Fulfil a Spare Part Request
- Restock a created Spare Part

All participants were able to get the expected result for each test step, which means no errors were found in the functional test.

Following was the usability test, which consisted of two parts. Firstly, the participants got six tasks with expected results. Their challenge was to reach the expected results with no description of each step. They also had the assignment to track the time for each task. Secondly, some questions regarding their overall satisfaction were asked. The tasks for the usability test were the following:

- Create a Machine with data and view this Machine afterwards
- Create a Ticket (with no Spare Part) with data and view this Ticket afterwards
- Create a ticket with data, request a Spare Part (with an amount) and view this Ticket afterwards
- Create a Spare Part with data and view this Spare Part
- Fulfil a Spare Part Request
- Restock a Spare Part

All participants were able to complete each task. The Results regarding the time it took them in comparison with the IT affinity of each participant, can be found in Figure 39. Interestingly, a trend can be seen, where our participants with a high IT affinity were able to complete the tasks faster than the ones with a lower IT knowledge. The last task in Figure 39 “Restock a Spare Part” took the participant with a high IT affinity level a lot longer, because he found a small bug, where a Spare Part couldn’t be restocked. On a side note, it has to be said, that the results rely on the limitation, that only five participants completed the test.

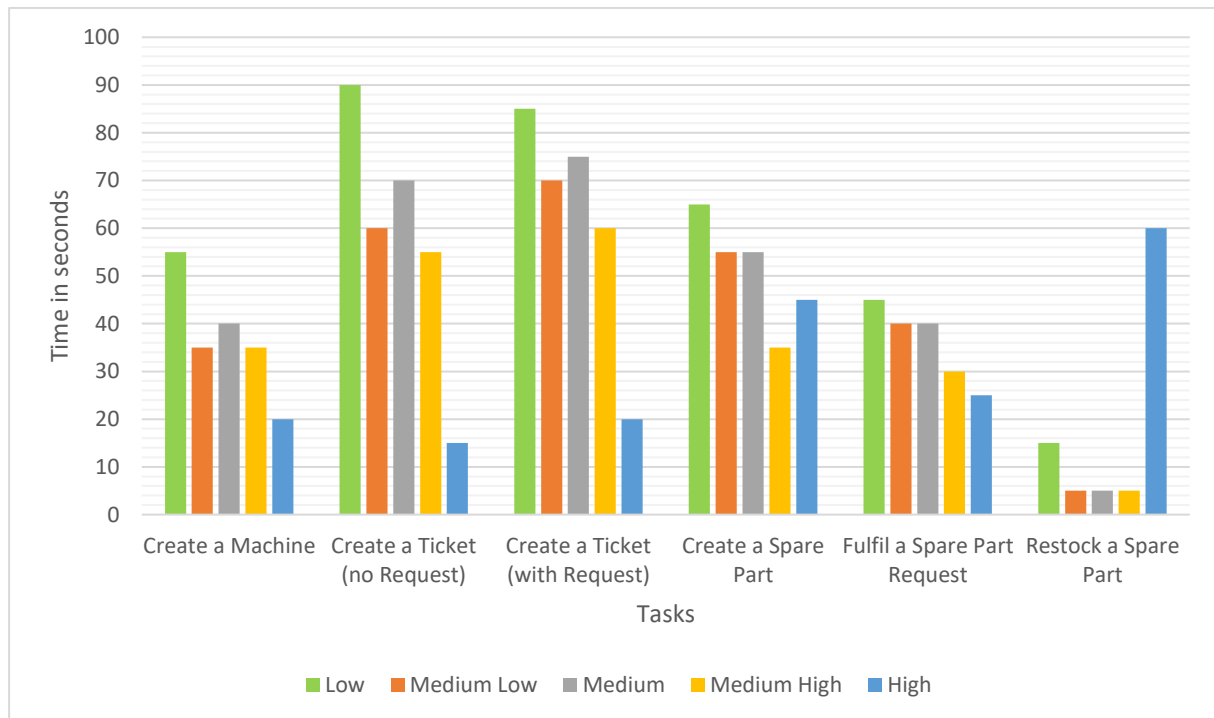


Figure 39 - Comparison of IT affinity and time on each task

After completing the first part of the usability test, the participants had to rate their satisfaction with the prototype by giving school grades (1 is best, 5 is worst) regarding:

- Ease of use
- User satisfaction
- Ease of finding information
- Visual design
- Navigation flow

The average of the results where a 1.88, where the biggest critic came from the visual design. The depiction of the information on some screens were not optimal. As an example: a participant with a smaller screen, added three machines with tickets, but the last machine was displayed on the left side, while the corresponding tickets were already displayed on the right side.

In conclusion the beta test was a success, a small bug was fixed and the installation guide was adapted with version numbers, so that the installation with an already existing NodeJS version will not cause any problems for future test users. The compact results regarding the defined metrics can be found in Table 35.

Table 35 - Overview of the metric results

Metric	Result
Successful task completion	All tasks were completed with success.
Errors	One small error was found regarding the restocking of a Spare Part.
Error-free rate	Since only one error was found the error free rate lies at about 99,97 percent.
Time on task	This result can be found in Figure 39. The average time on all tasks of the usability test for all participants were 4.5 minutes.
Subjective measures	The overall experience of all participants was good. The biggest potential for improvement is the visual design.

In the following chapter, a discussion and outlook are given, where the research questions are answered and future work is discussed.

5 Discussion & outlook

The last chapter of this thesis provides a discussion on the work of this thesis and an outlook to future research.

5.1 Discussion

In this chapter, the research questions will be discussed and the output regarding the problem statement will be analyzed. The research questions and their answers are the following:

- **What is an appropriate way of utilizing blockchain in smart manufacturing enterprises?**

A proof-of-concept for a blockchain solution for smart manufacturing was proposed and evaluated (cf. Chapter 5). The main characteristic of this blockchain is that it is a private blockchain. This limits the people, who can access information on the blockchain. In the chosen to be implemented use case, a public blockchain would interfere with the goal of data privacy. That is the reason why a private blockchain solution fits better.

Ethereum was the chosen blockchain for two reasons. The first reason is that it supports smart contracts. The foundation of the solution proposed in this thesis are smart contracts, since they represent the business logic. Through them the data model and functions such as getter and setter were defined. However, there are also other blockchains supporting smart contracts. The second reason why Ethereum was chosen is that it is further developed than other platforms. Additionally, the PoW consensus algorithm provides a more secure solution than others due to the validation process. There are also some restrictions to Ethereum, which will be discussed in the next chapter. Nevertheless, overall no major bugs or problems were encountered.

Smart contracts represent the business logic and data model. In this thesis four objects were defined, being a Machine, a Ticket, a Spare Part and a Spare Part Request. The functionality of those objects was mainly to set and get data of those objects in a secure, transparent and simple manner.

- **What are the industrial requirements for employing blockchain?**

The most important industrial requirement for the adoption of blockchain is a valid use-case. The question, why blockchain should be used and not a different technology, must be asked. The most interesting factors that must be considered when deciding are the following:

- Information security

- Decentralized storage
- Data immutability
- Smart contracts
- Data transparency

Therefore, if the use-case does not fulfil the requirements mentioned above, another solution without blockchain should be considered.

Furthermore, another requirement comes with the use of private blockchains. A business would need to employ distributed validation nodes, which constantly validate data incurring electricity and hardware costs.

In the chosen use case, storage is not a major issue, since only words and numbers are saved, but in other use cases the storage of a blockchain needs to be planned. This derives from the fact that the blockchain has to be stored on each node and, currently, no methods are archiving this.

- **What are the technological and non-technological risks associated to it?**

The main risks in adopting blockchain technology can be separated into two categories: risks arising from the use of a permissionless blockchain and from a permissioned blockchain. In this thesis, a permissioned blockchain was used to create the prototype.

Permissionless risks identified are open source code and crypto wallets or key management systems. Open source code works well for finding bugs and further development but can also be used maliciously. Key management systems or crypto wallets can be unsafe and one is dependent on the providers.

Permissioned blockchains operate in a more controlled environment but still have risks such as interoperability and standardization. Combining two permissioned blockchains would be a hard task, since they probably run a different data model and permission system. The lack of standardization can also be a risk if every company develops their own solution. The progress will then be slower and interoperability is not ensured.

Non-technological risks are mainly the unfamiliarity with the concept of blockchain and its possibilities. Most people who know blockchain only know its cryptocurrency aspect but not what else this technology is capable of. Another risk are costs, since blockchain is still in a very early stage of development and it is hard to say where the technological advancements will lead to.

The problem defined in chapter 1.2 was the need for a secure data sharing platform in an industrial environment. This thesis shows, that blockchain can provide this through its strong cryptologic characteristics. A comparison between a state-of-the-art example and the prototype proposed in this thesis showed that blockchain has many

mechanisms used by the state-of-the-art solution already built in. Features such as a watermark or access control are not required, or to be more precise already provided by blockchain. Timestamps and signing of transactions as well as a private blockchain, which only registered users can access, solves these issues.

5.2 Outlook

Smart manufacturing and blockchain are relatively new fields of research. Smart manufacturing includes various topics such as Industry 4.0, IoT, cloud, big data, smart factories, 3D printing and others. This thesis attempts to provide an overview of the possibilities in these fields. Nevertheless, there is room for more research. Due to the variety of technologies, there are many uses-cases where blockchain could address unsolved issues.

Moreover, blockchain is not too well researched. Although this technology has been around for some time now, other applications than in terms of cryptocurrency are brand new. Smart contracts and consensus algorithms are researched and further improved every day. For instance, changes in the structure of Solidity changes from update to update.

It is hard to say where to this journey leads, but if the disadvantages such as transaction times can be fixed, blockchain is a promising technology for more use cases especially in smart manufacturing and modern businesses in general.

Finally, some limitations of this work must be addressed. This thesis did not put any emphasis on scalability issues. On one hand, much research on this topic already exists and on the other hand, the amount of data processed is rather small. Additionally, problems occurred when more than 8 parameters were used in functions. The problem was solved, by splitting functions in two or three parts. This does not seem to be an efficient solution and future improvements on Solidity are necessary if use cases which require many parameters are needed, should be supported.

The proof-of-concept could be expanded by reading in IoT data automatically. This way, no manual work has to be done on the manufacturer's side. There are already some solutions for IoT and blockchain. A combination of one of those and the proof-of-concept should be investigated further. The concept of such a system can be seen in Figure 40. This adds IoT technology to the prototype developed in this thesis. Each IoT device represents a node in the blockchain and supplies it with machine data. Through that, machine problems could be detected automatically and the information could be saved in the blockchain.

In addition, a comparison between an Ethereum based solution and a solution using HyperLedger Fabric could be interesting due to its other characteristics. A better fit of

HyperLedger Fabric should be investigated, when introducing IoT into the concept developed in this thesis.

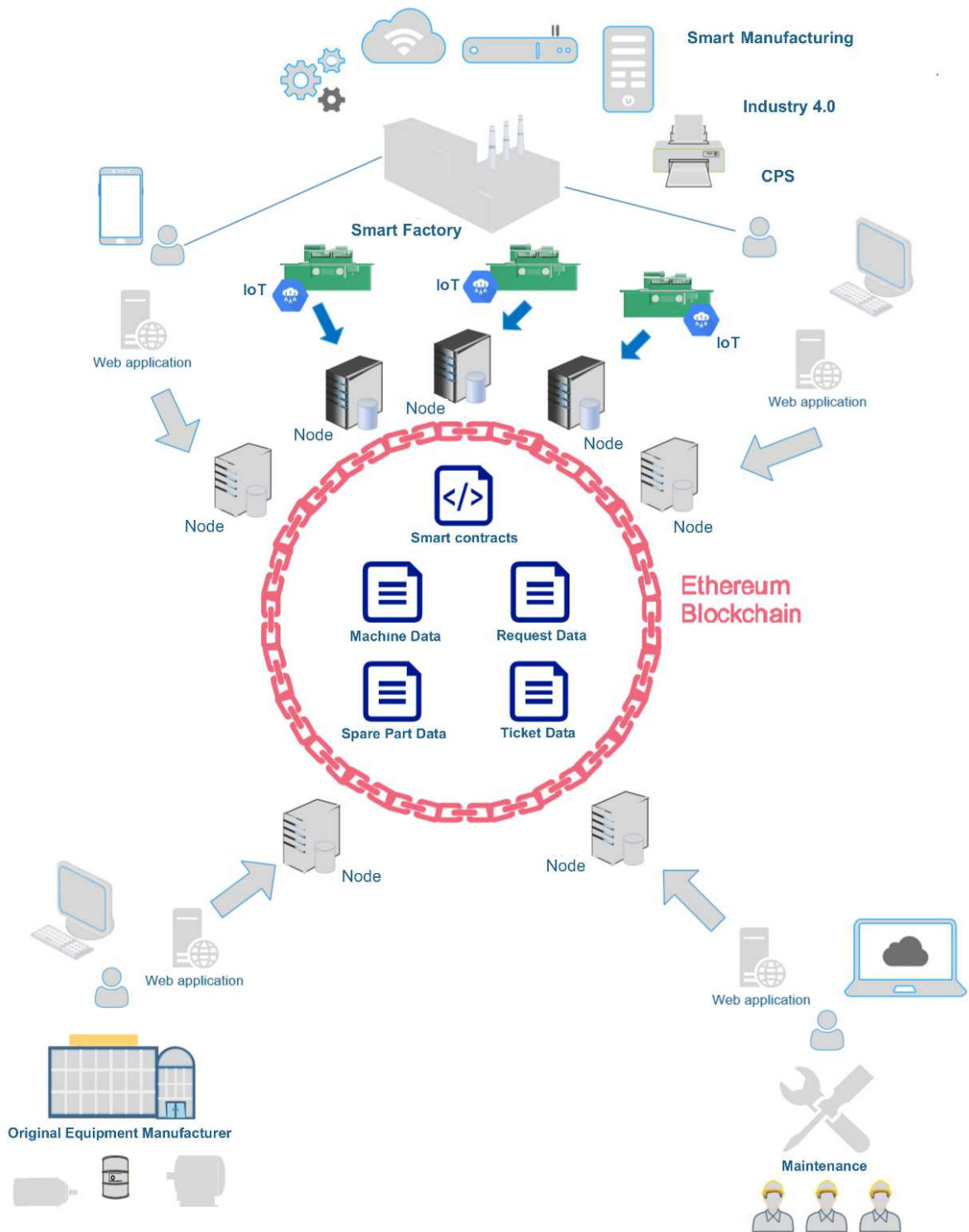


Figure 40 – Concept for a remote maintenance architecture using blockchain

6 Bibliography

Ahmed, I., Obermeier, S., Naedele, M., & Richard III, G. G. (2012). SCADA Systems: Challenges for Forensic Investigators. *Computer*, 45(12), 44–51, doi: 10.1109/MC.2012.325.

Allwood, J. M., Childs, T. H.C., Clare, A. T., Silva, A. K.M. de, Dhokia, V., Hutchings, I. M., et al. (2016). Manufacturing at double the speed. *Journal of Materials Processing Technology*, 229, 729–757, doi: 10.1016/j.jmatprotec.2015.10.028.

Bahga, A., & Madiseti, V. K. (2016). Blockchain Platform for Industrial Internet of Things. *Journal of Software Engineering and Applications*, 09(10), 533–546, doi: 10.4236/jsea.2016.910036.

Behl, A., & Behl, K. (2012). Security Paradigms for Cloud Computing. In G. Tomar, T. N. Sharma, & D. Bhatnagar (Eds.), *2012 4th International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN 2012), Phuket, Thailand, 7/24/2012 - 7/26/2012* (pp. 200–205). Los Alamitos, California: IEEE Computer Society, doi: 10.1109/CICSyN.2012.45.

Bishop, M. A. (2015). *Introduction to computer security*. Boston: Addison-Wesley.

Bizarro, P., Mankowski, R., & Mankowski, H. BLOCKCHAIN TECHNOLOGY: BENEFITS, RISKS, AND THE FUTURE. In *APA 6th - American Psychological Association* (pp. 12–16).

Bohn, R. B., Messina, J., Liu, F., Tong, J., & Mao, J. (2011). NIST Cloud Computing Reference Architecture. In *2011 IEEE World Congress on Services (SERVICES), Washington, DC, USA, 7/4/2011 - 7/9/2011* (pp. 594–596). Los Alamitos, Calif.: Conference Publishing Services, IEEE Computer Society, doi: 10.1109/SERVICES.2011.105.

Budak, E., Tunc, L. T., & Budak, E. (2011). Analytical methods for increased productivity in 5-axis ball-end milling. *International Journal of Mechatronics and Manufacturing Systems*, 4(3/4), 238, doi: 10.1504/IJMMS.2011.041471.

Budman, M., Hurley, B., Khan, A., & Gangopadhyaya, N. (2019). Deloitte's 2019 Global Blockchain Survey: Blockchain gets down to business.

Chen, B., Wan, J., Shu, L., Li, P., Mukherjee, M., & Yin, B. (2018). Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges. *IEEE Access*, 6, 6505–6519, doi: 10.1109/ACCESS.2017.2783682.

Dieterich, V., Ivanovic, M., Meier, T., Zäpfel, S., Utz, M., & Sandner, P. Application of Blockchain Technology in the Manufacturing Industry, 1–23.

Elhabashy, A. E., Wells, L. J., Camelio, J. A., & Woodall, W. H. (2019). A cyber-physical attack taxonomy for production systems: a quality control perspective. *Journal of Intelligent Manufacturing*, 30(6), 2489–2504, doi: 10.1007/s10845-018-1408-9.

(2019). FAQ - Bitcoin. <https://bitcoin.org/de/faq#was-ist-bitcoin>. Accessed 21 March 2019.

Fernández-Caramés, T. M., & Fraga-Lamas, P. (2019). *A Review on the Application of Blockchain for the Next Generation of Cybersecure Industry 4.0 Smart Factories*. <https://arxiv.org/pdf/1902.09604.pdf>. Accessed 10 May 2019.

Gatteschi, V., Lamberti, F., Demartini, C., Pranteda, C., & Santamaría, V. (2018). Blockchain and Smart Contracts for Insurance: Is the Technology Mature Enough? *Future Internet*, 10(2), doi: 10.3390/fi10020020.

Gökalp, E., Gökalp, M. O., Çoban, S., & Eren, P. E. (2018). Analysing Opportunities and Challenges of Integrated Blockchain Technologies in Healthcare. In S. Wrycza & J. Maślankowski (Eds.), *Information systems: Research, development, applications, education : 11th SIGSAND/PLAIS EuroSymposium 2018, Gdansk, Poland, September 20, 2018, Proceedings* (Vol. 333, pp. 174–183, Lecture Notes in Business Information Processing, Vol. 333). Cham, Switzerland: Springer.

Hermann, M., Pentek, T., & Otto, B. Design Principles for Industrie 4.0 Scenarios_logos.

Hevner, March, Park, & Ram (2004). Design Science in Information Systems Research. *MIS Quarterly*, 28(1), 75, doi: 10.2307/25148625.

Höltmann, A., & Ogyan, V. (2019). Wenn der Blockchain-Nebel sich lichtet, 1–12.

Hughes, A., Park, A., Kietzmann, J., & Archer-Brown, C. (2019). Beyond Bitcoin: What blockchain and distributed ledger technologies mean for firms. *Business Horizons*, doi: 10.1016/j.bushor.2019.01.002.

Hughes, L., Dwivedi, Y. K., Misra, S. K., Rana, N. P., Raghavan, V., & Akella, V. (2019). Blockchain research, practice and policy: Applications, benefits, limitations, emerging research themes and research agenda. *International Journal of Information Management*, 49, 114–129, doi: 10.1016/j.ijinfomgt.2019.02.005.

Hyvärinen, H., Risius, M., & Friis, G. (2017). A Blockchain-Based Approach Towards Overcoming Financial Fraud in Public Sector Services. *Business & Information Systems Engineering*, 59(6), 441–456, doi: 10.1007/s12599-017-0502-4.

(IEEE 2016). IEEE Standard for System, Software, and Hardware Verification and Validation. Piscataway, NJ, USA: IEEE, doi: 10.1109/IEEESTD.2017.8055462.

Kang, H. S., Lee, J. Y., Choi, S., Kim, H., Park, J. H., Son, J. Y., et al. (2016). Smart manufacturing: Past research, present findings, and future directions. *International Journal of Precision Engineering and Manufacturing-Green Technology*, 3(1), 111–128, doi: 10.1007/s40684-016-0015-5.

Krutz, R. L. (2006). *Securing SCADA systems*. Indianapolis, IN: Wiley Pub.

Laabs, M., & Đukanović, S. (2018). Blockchain in Industrie 4.0: Beyond cryptocurrency. *it - Information Technology*, 60(3), 143–153, doi: 10.1515/itit-2018-0011.

Lamberti, F., Gatteschi, V., Demartini, C., Pranteda, C., & Santamaria, V. (2017). Blockchain or not blockchain, that is the question of the insurance and other sectors. *IT Professional*, 1, doi: 10.1109/MITP.2017.265110355.

Lang, D., Friesen, M., Ehrlich, M., Wisniewski, L., & Jasperneite, J. (2018). Pursuing the Vision of Industrie 4.0: Secure Plug-and-Produce by Means of the Asset Administration Shell and Blockchain Technology. In *2018 IEEE 16th International Conference on Industrial Informatics (INDIN), Porto, 18.07.2018 - 20.07.2018* (pp. 1092–1097): IEEE, doi: 10.1109/INDIN.2018.8471939.

Lee, E. A., & Seshia, S. A. (2017). *Introduction to embedded systems: A cyber-physical systems approach*. Cambridge, Massachusetts, London, England: MIT Press.

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

Li, Q., Jiang, H., Tang, Q., Chen, Y., Li, J., & Zhou, J. (2017). Smart Manufacturing Standardization: Reference Model and Standards Framework. In I. Ciuciu, C. Debruyne, H. Panetto, G. Weichhart, P. Bollen, A. Fensel, et al. (Eds.), *On the Move to Meaningful Internet Systems: Otm 2016 Workshops - Confederated International Workshops, Revised Selected Papers* (Vol. 10034, pp. 16–25, Lecture Notes in Computer Science). Cham: Springer-Verlag New York Inc.

Livesey, N. (2016). Future-of-Manufacturing-Winter-2016. <https://www.pinsentmasons.com/PDF/2016/Future-of-Manufacturing-Winter-2016.pdf>. Accessed 28 March 2019.

Luu, L., Chu, D.-H., Olickel, H., Saxena, P., & Hobor, A. (2016). Making Smart Contracts Smarter. In S. Katzenbeisser & E. Weippl (Eds.), *the 2016 ACM SIGSAC Conference, Vienna, Austria, 24.10.2016 - 28.10.2016* (pp. 254–269). New York, NY: Association for Computing Machinery, doi: 10.1145/2976749.2978309.

Manufacturing Lounge (2018). How are smart factories changing the face of manufacturing? - Manufacturing Lounge. <http://www.manufacturinglounge.com/how-are-smart-factories-changing-the-face-of-manufacturing/>. Accessed 5 April 2019.

Martynov, V. V., Shavaleeva, D. N., & Zaytseva, A. A. (2019). Information Technology as the Basis for Transformation into a Digital Society and Industry 5.0. In *2019 International Conference "Quality Management, Transport and Information Security, Information Technologies" (IT&QM&IS), Sochi, Russia, 9/23/2019 - 9/27/2019* (pp. 539–543): IEEE, doi: 10.1109/ITQMIS.2019.8928305.

Mendling, J., Dustdar, S., Gal, A., García-Bañuelos, L., Governatori, G., Hull, R., et al. (2018). Blockchains for Business Process Management - Challenges and Opportunities. *ACM Transactions on Management Information Systems*, 9(1), 1–16, doi: 10.1145/3183367.

Michela, M. (2018). Is Blockchain A Viable Technology For Industry 4.0? - ProQuest. <https://search.proquest.com/docview/2050068035?accountid=39579>. Accessed 20 April 2019.

Mohamed, N., & Al-Jaroodi, J. (2019). Applying Blockchain in Industry 4.0 Applications. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 1/7/2019 - 1/9/2019* (pp. 852–858): IEEE, doi: 10.1109/CCWC.2019.8666558.

Monostori, L., Kádár, B., Bauernhansl, T., Kondoh, S., Kumara, S., Reinhart, G., et al. (2016). Cyber-physical systems in manufacturing. *CIRP Annals*, 65(2), 621–641, doi: 10.1016/j.cirp.2016.06.005.

Moon, I., Lee, G. M., Park, J., Kiritsis, D., & Cieminski, G. von (Eds.) (2018). *Advances in Production Management Systems. Smart Manufacturing for Industry 4.0: IFIP WG 5.7 International Conference, APMS 2018, Seoul, Korea, August 26-30, 2018, Proceedings, Part II* (IFIP Advances in Information and Communication Technology, Vol. 536). Cham: Springer International Publishing.

NIST (2017). Smart Manufacturing Operations Planning and Control Program. <https://www.nist.gov/programs-projects/smart-manufacturing-operations-planning-and-control-program>. Accessed 28 March 2019.

Nofer, M., Gomber, P., Hinz, O., & Schiereck, D. (2017). Blockchain, 183–187, doi: 10.1007/s12599-017-0467-3.

Ouyang, Y.-x., Tang, M., Lin, J.-c., & Dong, J.-x. (2004). Distributed collaborative CAD system based on Web Service. *Journal of Zhejiang University. Science*, 5(5), 579–586, doi: 10.1631/jzus.2004.0579.

- Panarello, A., Tapas, N., Merlino, G., Longo, F., & Puliafito, A. (2018). Blockchain and IoT Integration: A Systematic Survey. *Sensors (Basel, Switzerland)*, 18(8), 1–37, doi: 10.3390/s18082575.
- Park, J.-S., Youn, T.-Y., Kim, H.-B., Rhee, K.-H., & Shin, S.-U. (2018). Smart Contract-Based Review System for an IoT Data Marketplace. *Sensors (Basel, Switzerland)*, 18(10), doi: 10.3390/s18103577.
- Polyzos, G. C., & Fotiou, N. (2017). Blockchain-Assisted Information Distribution for the Internet of Things. In C. Zhang (Ed.), *2017 IEEE International Conference on Information Reuse and Integration (IRI), San Diego, CA, 8/4/2017 - 8/6/2017* (pp. 75–78). Piscataway, NJ: IEEE, doi: 10.1109/IRI.2017.83.
- Püttgen, F., & Kaulartz, M. (2017). Versicherung 4.0. *ERA Forum*, 18(2), 249–262, doi: 10.1007/s12027-017-0479-y.
- Rajput, S., & Singh, S. P. (2019). Industry 4.0 – challenges to implement circular economy. *Benchmarking: An International Journal*, 2(10), 1, doi: 10.1108/BIJ-12-2018-0430.
- Rittinghouse, J. W., & Ransome, J. F. (2016). *Cloud computing: Implementation, management, and security*. Boca Raton, FL: CRC Press.
- Rutledge, L. S., & Hoffman, L. J. (1986). A survey of issues in computer network security. *Computers & Security*, 5(4), 296–308, doi: 10.1016/0167-4048(86)90050-7.
- Sara Grasel (2018). So funktioniert Österreichs erster Ehevertrag auf der Blockchain. <https://www.trendingtopics.at/so-funktioniert-oesterreichs-erster-ehevertrag-auf-der-blockchain/>. Accessed 28 May 2019.
- Schoder, D., & Fischbach, K. (2003). Peer-to-peer prospects. *Communications-of the ACM*,. *Wirtschaftsinformatik*, 46(2), 27–29.
- Schoder, D., Fischbach, K., & Schmitt, C. (2004). Core Concepts in Peer-to-Peer Networking. In *Peer-to-Peer Computing* : IGI Global.
- Shrier, D., Wu, W., & Pentland, A. (2016). *Blockchain & Infrastructure (Identity, Data Security)*, 1–18.
- Thames, L., & Schaefer, D. (2017). *Cybersecurity for Industry 4.0*. Cham: Springer International Publishing.
- Tuptuk, N., & Hailes, S. (2018). Security of smart manufacturing systems. *Journal of Manufacturing Systems*, 47, 93–106, doi: 10.1016/j.jmsy.2018.04.007.
- TÜV AUSTRIA, & Fraunhofer Austria (2018). *Safety & Security in der Mensch-Roboter-Kollaboration: Mensch-RoboterKollaboration*. Accessed 10 September 2019.

Udokwu, C., Aleksandr Kormiltsyn, Kondwani Thangalimodzi, & Alex Norta. *An Exploration of Blockchain enabled Smart-Contracts Application in the Enterprise* .

Underwood, S. (2016). Blockchain beyond bitcoin. *Communications of the ACM*, 59(11), 15–17, doi: 10.1145/2994581.

Verein Industrie 4.0 Österreich. Forschung, Entwicklung & Innovation in der Industrie 4.0, 2018.

Verein Industrie 4.0 Österreich (2019). Industry 4.0. <https://plattformindustrie40.at/was-ist-industrie-4-0/?lang=en>. Accessed 28 March 2019.

Viriyasitavat, W., Xu, L. D., Bi, Z., & Sapsomboon, A. (2018). Correction to: Blockchain-based business process management (BPM) framework for service composition in industry 4.0. *Journal of Intelligent Manufacturing*, doi: 10.1007/s10845-018-1449-0.

Vukolić, M. (2016). The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication. In J. Camenisch & D. Kesdoğan (Eds.), *Open problems in network security* (Vol. 9591, pp. 112–125, Lecture notes in computer science Security and cryptology, Vol. 9591). Cham, Heidelberg: Springer.

WinterGreen Research (2018). Blockchain Market Shares, Market Strategies, and Market Forecasts, 2018 to 2024.

Wu, D., Rosen, D. W., & Schaefer, D. (2014). Cloud-Based Design and Manufacturing: Status and Promise. In D. Schaefer (Ed.), *Cloud-based design and manufacturing (CBDM): A service-oriented product development paradigm for the 21st century* (Vol. 7, pp. 1–24). Berlin: Springer.

Wu, D., Thames, J. L., Rosen, D. W., & Schaefer, D. (2012). Towards a Cloud-Based Design and Manufacturing Paradigm: Looking Backward, Looking Forward. In *ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Chicago, Illinois, USA, 8/12/2012 - 8/15/2012* (pp. 315–328). [S.I.]: ASME, doi: 10.1115/DETC2012-70780.

Wu, Y., HE, F., Li, W., CAI, X., & LI, X. (2016). Sensitive Information Protection of CAD Model Based on Free-Form Deformation in Collaborative Design. In Q. Zu & B. Hu (Eds.), *Human centered computing: Second international conference, hcc 2016, colombo* (Vol. 9567, pp. 465–474, Lecture Notes in Computer Science). [Place of publication not identified]: Springer.

Xing, B., & Marwala, T. (2018). *Smart Maintenance for Human–Robot Interaction* (Vol. 129). Cham: Springer International Publishing.

Xu, L. D., Xu, E. L., & Li, L. (2018). Industry 4.0: state of the art and future trends. *International Journal of Production Research*, *56*(8), 2941–2962, doi: 10.1080/00207543.2018.1444806.

Yao, X., Zhou, J., Lin, Y., Li, Y., Yu, H., & Liu, Y. (2017). Smart manufacturing based on cyber-physical systems and beyond. *Journal of Intelligent Manufacturing*, *10*(1), 1–13, doi: 10.1007/s10845-017-1384-5.

Zamani, E., He, Y., & Phillips, M. (2018). On the Security Risks of the Blockchain. *Journal of Computer Information Systems*, *90*(2), 1–12, doi: 10.1080/08874417.2018.1538709.

Zheng, P., wang, H., Sang, Z., Zhong, R. Y., Liu, Y., Liu, C., et al. (2018). Smart manufacturing systems for Industry 4.0: Conceptual framework, scenarios, and future perspectives. *Frontiers of Mechanical Engineering*, *13*(2), 137–150, doi: 10.1007/s11465-018-0499-5.

7 Attachment

7.1 Smart contract

7.1.1 Machine

```
1. pragma solidity ^0.5.1;
2.
3. contract MachineContract{
4.
5.     struct Machine{
6.         address machineNr;
7.         uint256 totalMachineWeight;
8.         string dimension;
9.         uint256 colour;
10.        string ambientTemperature;
11.        string storageTemperature;
12.        uint256 installationAltitudeMax;
13.        string relativeHumidity;
14.        uint256 powerVoltage;
15.        string typeOfPower;
16.        string frequency;
17.        uint256 installedPower;
18.        uint256 nominalCurrent;
19.        address[] tickets;
20.        string status;
21.        string comment;
22.        uint256 cutOffPower;
23.        string wireCrossSectionArea;
24.        string auxiliaryVoltage;
25.        string CNCPowerSupplyVoltage;
26.        string enclosureProtectionDegree;
27.        uint256 gTankCapacity;
28.        uint256 gPumpFlowRate;
29.        string gPressure;
30.        uint256 mTankCapacity;
31.        string mFlowRate;
32.        string mAverageConsumption;
33.        string bendingCapacity;
34.        uint256 xAxisShiftSpeed;
35.        uint256 yAxisBendingSpeed;
36.        uint256 zAxisRotationSpeed;
37.        uint256 qAxisPositioningSpeed;
38.        uint256 rAxisPositioningSpeed;
39.        string tolerances;
40.        uint256 maximumAverageProduction;
41.        uint256 maxDieWeight;
42.    }
43.
44.    mapping (address => Machine) machineList;
45.    address[] machineData;
46.
47.    /*set machine data*/
48.    function setMachineData1(address _machineNr,
49.        uint256 _totalMachineWeight,
50.        string memory _dimension,
51.        uint256 _colour,
52.        string memory _ambientTemperature,
53.        string memory _storageTemperature,
54.        uint256 _installationAltitudeMax,
55.        string memory _status,
56.        string memory _comment
```

```
57.         ) public{
58.     Machine storage machine = machineList[_machineNr];
59.
60.     machine.machineNr = _machineNr;
61.     machine.totalMachineWeight = _totalMachineWeight;
62.     machine.dimension = _dimension;
63.     machine.colour = _colour;
64.     machine.ambientTemperature = _ambientTemperature;
65.     machine.storageTemperature = _storageTemperature;
66.     machine.installationAltitudeMax = _installationAltitudeMax;
67.     machine.status = _status;
68.     machine.comment = _comment;
69.
70.     machineData.push(_machineNr) -1;
71. }
72.
73. function setMachineData2(address _machineNr,
74.     string memory _relativeHumidity,
75.     uint256 _powerVoltage,
76.     string memory _typeOfPower,
77.     string memory _frequency,
78.     uint256 _installedPower,
79.     uint256 _nominalCurrent
80.     ) public {
81.     Machine storage machine = machineList[_machineNr];
82.
83.     machine.machineNr = _machineNr;
84.     machine.relativeHumidity = _relativeHumidity;
85.     machine.powerVoltage = _powerVoltage;
86.     machine.typeOfPower = _typeOfPower;
87.     machine.frequency = _frequency;
88.     machine.installedPower = _installedPower;
89.     machine.nominalCurrent = _nominalCurrent;
90. }
91.
92. function setMachineData3(address _machineNr,
93.     uint256 _cutOffPower,
94.     string memory _wireCrossSectionArea,
95.     string memory _auxiliaryVoltage,
96.     string memory _CNCPowerSupplyVoltage,
97.     string memory _enclosureProtectionDegree,
98.     uint256 _gTankCapacity,
99.     uint256 _gPumpFlowRate,
100.    string memory _gPressure
101.    ) public {
102.    Machine storage machine = machineList[_machineNr];
103.
104.    machine.machineNr = _machineNr;
105.    machine.cutOffPower = _cutOffPower;
106.    machine.wireCrossSectionArea = _wireCrossSectionArea;
107.    machine.auxiliaryVoltage = _auxiliaryVoltage;
108.    machine.CNCPowerSupplyVoltage = _CNCPowerSupplyVoltage;
109.    machine.enclosureProtectionDegree = _enclosureProtectionDegree;
110.    machine.gTankCapacity = _gTankCapacity;
111.    machine.gPumpFlowRate = _gPumpFlowRate;
112.    machine.gPressure = _gPressure;
113. }
114.
115. function setMachineData4(address _machineNr,
116.     uint256 _mTankCapacity,
117.     string memory _mFlowRate,
118.     string memory _mAverageConsumption,
119.     string memory _bendingCapacity,
120.     uint256 _xAxisShiftSpeed,
121.     uint256 _yAxisBendingSpeed,
122.     uint256 _zAxisRotationSpeed
123.     ) public {
```

```

124.         Machine storage machine = machineList[_machineNr];
125.
126.         machine.machineNr = _machineNr;
127.         machine.mTankCapacity = _mTankCapacity;
128.         machine.mFlowRate = _mFlowRate;
129.         machine.mAverageConsumption = _mAverageConsumption;
130.         machine.bendingCapacity = _bendingCapacity;
131.         machine.xAxisShiftSpeed = _xAxisShiftSpeed;
132.         machine.yAxisBendingSpeed = _yAxisBendingSpeed;
133.         machine.zAxisRotationSpeed = _zAxisRotationSpeed;
134.     }
135.
136.     function setMachineData5(address _machineNr,
137.                             uint256 _qAxisPositioningSpeed,
138.                             uint256 _rAxisPositioningSpeed,
139.                             string memory _tolerances,
140.                             uint256 _maximumAverageProduction,
141.                             uint256 _maxDieWeight
142.                             ) public {
143.         Machine storage machine = machineList[_machineNr];
144.
145.         machine.machineNr = _machineNr;
146.         machine.qAxisPositioningSpeed = _qAxisPositioningSpeed;
147.         machine.rAxisPositioningSpeed = _rAxisPositioningSpeed;
148.         machine.tolerances = _tolerances;
149.         machine.maximumAverageProduction = _maximumAverageProduction;
150.         machine.maxDieWeight = _maxDieWeight;
151.     }
152.
153.
154.     /*get Machine data*/
155.     function getMachineData1(address _machineNr) public view returns (uint256
156.         _totalMachineWeight,
157.         string memory _dimension,
158.         uint256 _colour,
159.         string memory _ambientTemperature,
160.         string memory _storageTemperature,
161.         uint256 _installationAltitudeMax){
162.         return (machineList[_machineNr].totalMachineWeight, machineList[_
163.         machineNr].dimension, machineList[_machineNr].colour,
164.         machineList[_machineNr].ambientTemperature, machineList[_machineNr].s
165.         torageTemperature, machineList[_machineNr].installationAltitudeMax);
166.     }
167.
168.     function getMachineData2(address _machineNr) public view returns (string
169.         memory _relativeHumidity,
170.         uint256 _powerVoltage,
171.         string memory _typeOfPower,
172.         string memory _frequency,
173.         uint256 _installedPower,
174.         uint256 _nominalCurrent){
175.         return (machineList[_machineNr].relativeHumidity, machineList[_machin
176.         eNr].powerVoltage, machineList[_machineNr].typeOfPower, machineList[_machineNr].freq
177.         uency,
178.         machineList[_machineNr].installedPower, machineList[_machineNr].nomin
179.         alCurrent);
180.     }
181.
182.     function getMachineData3(address _machineNr) public view returns (uint256
183.         _cutOffPower,
184.         string memory _wireCrossSectionArea,
185.         string memory _auxiliaryVoltage,
186.         string memory _CNCPowerSupplyVoltage,
187.         string memory _enclosureProtectionDegree,
188.         uint256 _gTankCapacity,
189.         uint256 _gPumpFlowRate
190.         ){

```

```
183.         return (machineList[_machineNr].cutOffPower, machineList[_machineNr].
    wireCrossSectionArea, machineList[_machineNr].auxiliaryVoltage, machineList[_machine
    Nr].CNCPowerSupplyVoltage,
184.             machineList[_machineNr].enclosureProtectionDegree, machineList[_machi
    neNr].gTankCapacity, machineList[_machineNr].gPumpFlowRate);
185.     }
186.
187.     function getMachineData4(address _machineNr) public view returns (string
    memory _gPressure,
188.         uint256 _mTankCapacity,
189.         string memory _mFlowRate,
190.         string memory _mAverageConsumption,
191.         string memory _bendingCapacity,
192.         uint256 _xAxisShiftSpeed){
193.         return (machineList[_machineNr].gPressure, machineList[_machineNr].mT
    ankCapacity, machineList[_machineNr].mFlowRate, machineList[_machineNr].mAverageCons
    umption, machineList[_machineNr].bendingCapacity, machineList[_machineNr].xAxisShift
    Speed);
194.     }
195.
196.     function getMachineData5(address _machineNr) public view returns ( uint2
    56 _yAxisBendingSpeed,
197.         uint256 _zAxisRotationSpeed,
198.         uint256 _qAxisPositioningSpeed,
199.         uint256 _rAxisPositioningSpeed,
200.         string memory _tolerances,
201.         uint256 _maximumAverageProduction,
202.         uint256 _maxDieWeight){
203.         return (machineList[_machineNr].yAxisBendingSpeed, machineList[_machi
    neNr].zAxisRotationSpeed,
204.             machineList[_machineNr].qAxisPositioningSpeed, machineList[_machineNr
    ].rAxisPositioningSpeed, machineList[_machineNr].tolerances, machineList[_machineNr]
    .maximumAverageProduction, machineList[_machineNr].maxDieWeight);
205.     }
206.
207.     function countMachines() view public returns (uint) {
208.         return machineData.length;
209.     }
210.
211.     function getMachineAdress() view public returns (address[] memory) {
212.         return machineData;
213.     }
214.
215.     function setTicket(address _machineNr, address _address) public {
216.         Machine storage machine = machineList[_machineNr];
217.
218.         machine.tickets.push(_address) -1;
219.     }
220.
221.
222.     function getTicketAdress(address _machineNr) view public returns (address
    [] memory) {
223.         Machine storage machine = machineList[_machineNr];
224.
225.         return machine.tickets;
226.     }
227.
228.     function getMachineStatus(address _machineNr) view public returns (string
    memory) {
229.         Machine storage machine = machineList[_machineNr];
230.
231.         return machine.status;
232.     }
233.
234.     function setMachineStatus(address _machineNr, string memory _status) publ
    ic{
235.         Machine storage machine = machineList[_machineNr];
```

```

236.         machine.status = _status;
237.     }
238.
239.     function getMachineComment(address _machineNr) view public returns (string memory) {
240.         Machine storage machine = machineList[_machineNr];
241.
242.         return machine.comment;
243.     }
244.
245.     function setMachineComment(address _machineNr, string memory _comment) public {
246.         Machine storage machine = machineList[_machineNr];
247.         machine.comment = _comment;
248.     }
249. }

```

7.1.2 Ticket

```

1. pragma solidity ^0.5.1;
2.
3. contract TicketContract {
4.
5.     struct Ticket {
6.         address ticketNr;
7.         string interventionString;
8.         string interventionTypes;
9.         string machineType;
10.        string model;
11.        string unit;
12.        string operation;
13.        string cardCategories;
14.        uint256 frequency;
15.        uint256 time;
16.        bool machineStatus;
17.        uint256 operator;
18.        string sparePart;
19.        string specificTool;
20.    }
21.
22.    mapping (address => Ticket) ticketList;
23.    address[] ticketData;
24.
25.    event saveTicket1(
26.        string interventionString,
27.        string interventionTypes,
28.        string machineType,
29.        string model,
30.        string unit,
31.        string operation,
32.        string cardCategories
33.    );
34.
35.    event saveTicket2(
36.        uint256 frequency,
37.        uint256 time,
38.        bool machineStatus,
39.        uint256 operator,
40.        string sparePart,
41.        string specificTool
42.    );
43.
44.
45.
46.    function setTicketData1( address _ticketNr,

```

```

47.         string memory _interventionString,
48.         string memory _interventionTypes,
49.         string memory _machineType,
50.         string memory _model,
51.         string memory _unit,
52.         string memory _operation,
53.         string memory _cardCategories
54.     ) public returns(bool) {
55.
56.         Ticket storage ticket = ticketList[_ticketNr];
57.         ticket.ticketNr = _ticketNr;
58.         ticket.interventionString = _interventionString;
59.         ticket.interventionTypes = _interventionTypes;
60.         ticket.machineType = _machineType;
61.         ticket.model = _model;
62.         ticket.unit = _unit;
63.         ticket.operation = _operation;
64.         ticket.cardCategories = _cardCategories;
65.
66.         ticketData.push(_ticketNr) -1;
67.
68.         emit saveTicket1(_interventionString, _interventionTypes, _machineType,
        _model, _unit, _operation, _cardCategories);
69.
70.         return true;
71.     }
72.
73.
74.     function setTicketData2( address _ticketNr,
75.                             uint256 _frequency,
76.                             uint256 _time,
77.                             bool _machineStatus,
78.                             uint256 _operator,
79.                             string memory _sparePart,
80.                             string memory _specificTool
81.     ) public returns(bool) {
82.
83.         Ticket storage ticket = ticketList[_ticketNr];
84.         ticket.frequency = _frequency;
85.         ticket.time = _time;
86.         ticket.machineStatus = _machineStatus;
87.         ticket.operator = _operator;
88.         ticket.sparePart = _sparePart;
89.         ticket.specificTool = _specificTool;
90.
91.         emit saveTicket2(_frequency, _time, _machineStatus, _operator, _sparePar
t, _specificTool);
92.
93.         return true;
94.     }
95.
96.
97.     /*get Ticket data*/
98.     function getTicketData1(address _ticketNr) public view returns ( string memo
ry _interventionString,
99.     string memory _interventionTypes,
100.     string memory _machineType,
101.     string memory _model,
102.     string memory _unit,
103.     string memory _operation,
104.     string memory _cardCategories){
105.         return (ticketList[_ticketNr].interventionString, ticketList[_tic
ketNr].interventionTypes,
106.         ticketList[_ticketNr].machineType, ticketList[_ticketNr].model, t
icketList[_ticketNr].unit,
107.         ticketList[_ticketNr].operation, ticketList[_ticketNr].cardCatego
ries);

```

```

108.         }
109.
110.         function getTicketData2(address _ticketNr) public view returns ( uint
111.             256 _frequency,
112.             uint256 _time,
113.             bool _machineStatus,
114.             uint256 _operator,
115.             string memory _sparePart,
116.             string memory _specificTool){
117.             return (ticketList[_ticketNr].frequency, ticketList[_ticketNr].ti
118.                 me,
119.                 ticketList[_ticketNr].machineStatus, ticketList[_ticketNr].operat
120.                 or, ticketList[_ticketNr].sparePart, ticketList[_ticketNr].specificTool);
121.         }
122.
123.         function countTickets() view public returns (uint) {
124.             return ticketData.length;
125.         }
126.
127.         function getTicket(uint id) view public returns (address) {
128.             return ticketData[id];
129.         }
130.     }

```

7.1.3 Spare Part

```

1. pragma solidity ^0.5.1;
2.
3. contract SparePartContract {
4.     struct SparePart {
5.         address sparPartNr;
6.         string name;
7.         string description;
8.         bool status;
9.         uint256 amount;
10.    }
11.    mapping (address => SparePart) sparePartList;
12.    address[] sparePartData;
13.
14.    function setSparePartData1( address _sparePartNr,
15.        string memory _name,
16.        string memory _description,
17.        bool _status,
18.        uint256 _amount
19.    ) public {
20.
21.        SparePart storage sparePart = sparePartList[_sparePartNr];
22.        sparePart.name = _name;
23.        sparePart.description = _description;
24.        sparePart.status = _status;
25.        sparePart.amount = _amount;
26.
27.        sparePartData.push(_sparePartNr) -1;
28.    }
29.    function getSparePartData1(address _sparePartNr) public view returns (string mem
30.        ory _name,
31.        string memory _description,
32.        bool _status,
33.        uint256 _amount){
34.        return (sparePartList[_sparePartNr].name, sparePartList[_sparePartNr].descri
35.            ption, sparePartList[_sparePartNr].status, sparePartList[_sparePartNr].amount);
36.    }
37.    function setStatus( address _sparePartNr,
38.        bool _status
39.    ) public {

```

```

38.         SparePart storage sparePart = sparePartList[_sparePartNr];
39.         sparePart.status = _status;
40.     }
41.     function setAmount( address _sparePartNr,
42.                         uint256 _amount
43.                     ) public {
44.         SparePart storage sparePart = sparePartList[_sparePartNr];
45.         sparePart.amount = _amount;
46.     }
47.     function getAmount(address _sparePartNr) public view returns (uint256 _amount){
48.         return (sparePartList[_sparePartNr].amount);
49.     }
50.     function countSpareparts() view public returns (uint) {
51.         return sparePartData.length;
52.     }
53.     function getSparepartAddress() view public returns (address[] memory) {
54.         return sparePartData;
55.     }
56. }

```

7.1.4 Spare Part Request

```

1. pragma solidity ^0.5.1;
2.
3. contract SparePartRequestContract {
4.
5.     struct SparePartRequest {
6.         address sparePartRequestNr;
7.         string name;
8.         string description;
9.         address machineAddress;
10.        address ticketAddress;
11.        bool status;
12.        uint256 amount;
13.    }
14.
15.    mapping (address => SparePartRequest) sparePartRequestList;
16.    address[] sparePartRequestData;
17.
18.
19.    function setSparePartRequestData1( address _sparePartRequestNr,
20.                                       string memory _name,
21.                                       string memory _description,
22.                                       address _machineAddress,
23.                                       address _ticketAddress,
24.                                       bool _status,
25.                                       uint256 _amount
26.                                   ) public {
27.
28.        SparePartRequest storage sparePartRequest = sparePartRequestList[_spareP
artRequestNr];
29.        sparePartRequest.name = _name;
30.        sparePartRequest.description = _description;
31.        sparePartRequest.machineAddress = _machineAddress;
32.        sparePartRequest.ticketAddress = _ticketAddress;
33.        sparePartRequest.status = _status;
34.        sparePartRequest.amount = _amount;
35.
36.        sparePartRequestData.push(_sparePartRequestNr) -1;
37.    }
38.

```



```

39.     function getSparePartRequestData1(address _sparePartRequestNr) public view returns (string memory _name,
40. string memory _description,
41. address _machineAddress,
42. address _ticketAddress,
43. bool _status){
44.     return (sparePartRequestList[_sparePartRequestNr].name, sparePartRequestList[_sparePartRequestNr].description, sparePartRequestList[_sparePartRequestNr].machineAddress, sparePartRequestList[_sparePartRequestNr].ticketAddress, sparePartRequestList[_sparePartRequestNr].status);
45.     }
46.
47.     function setStatus( address _sparePartRequestNr,
48. bool _status
49. ) public {
50.
51.     SparePartRequest storage sparePartRequest = sparePartRequestList[_sparePartRequestNr];
52.     sparePartRequest.status = _status;
53.     }
54.
55.     function setAmount( address _sparePartRequestNr,
56. uint256 _amount
57. ) public {
58.
59.     SparePartRequest storage sparePartRequest = sparePartRequestList[_sparePartRequestNr];
60.     sparePartRequest.amount = _amount;
61.     }
62.
63.     function getAmount(address _sparePartRequestNr) public view returns (uint256 _amount){
64.     return (sparePartRequestList[_sparePartRequestNr].amount);
65.     }
66.
67.     function countSparePartRequests() view public returns (uint) {
68.     return sparePartRequestData.length;
69.     }
70.
71.     function getSparePartRequestAddress() view public returns (address[] memory) {
72.     return sparePartRequestData;
73.     }
74. }

```

7.2 Front end

7.2.1 Creating a Spare Part

```

1. <!DOCTYPE html>
2. <html lang="en">
3.
4. <head>
5.     <meta charset="UTF-8">
6.     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7.     <meta http-equiv="X-UA-Compatible" content="ie=edge">
8.     <title>Document</title>
9.     <link rel="stylesheet" type="text/css" href="main.css">
10.    <script src="./node_modules/web3/dist/web3.min.js"></script>
11. </head>
12.
13. <body>
14.     <div class="container">
15.         <div id="newSparePart">

```

```

16.         <h1>Spare Part</h1>
17.         <span id="countSparePart"></span>
18.
19.         <h2></h2>
20.         <span id="insTrans"></span>
21.         <hr>
22.
23.         
24.
25.         <label for="name" class="col-lg-2 control-label">name</label>
26.         <input id="name" type="text">
27.
28.         <label for="description" class="col-lg-2 control-
label">description</label>
29.         <input id="description" type="text">
30.
31.         <label for="amount" class="col-lg-2 control-label">amount</label>
32.         <input id="amount" type="number">
33.     </div>
34.     <div id="requestDiv">
35.         <hr>
36.         <label for="sparePart" class="col-lg-2 control-
label">Spare Part</label>
37.         <select id="sparePart">
38.         </select>
39.         <br><br>
40.
41.         <label for="amount2" class="col-lg-2 control-label">Amount</label>
42.         <input id="amount2" type="number">
43.     </div>
44.     <button id="button">Save</button>
45.     <button onclick="window.location.href='Index_Spare.html'">Back</button>
46. </div>
47.
48. <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
49.
50. <script>
51.     if (typeof web3 !== 'undefined') {
52.         web3 = new Web3(web3.currentProvider);
53.     } else {
54.         // set the provider you want from Web3.providers
55.         web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"))
56.     };
57.     web3web3.eth.defaultAccount = web3.eth.accounts[0];
58.
59.     //Define ABIs
60.     var TicketContract = web3.eth.contract(...ABI...);
61.     var MachineContract = web3.eth.contract(...ABI...);
62.     var SparePartContract = web3.eth.contract(...ABI...);
63.     var SparePartRequestContract = web3.eth.contract(...ABI...);
64.
65.     //Define contracts
66.     var Ticket = TicketContract.at('0x7e06ccb5c46db5ed809d9d8f73132608495f894c')
67.     ;
68.     var Machine = MachineContract.at('0x7d51587098e47cbc2b3467e2e1eb3ab4a49650ad
69.     6396');
70.     var SparePart = SparePartContract.at('0x82e4fb55e67dd09caa29ea1a7dc283180c13
71.     5dfc7587aa04eb459c');
72.
73.     var parameters = new URLSearchParams(window.location.search);
74.     var sparePartsList = new Array();
75.
76.     if (parameters.get('sprId') == null) { //Add Spare part

```

```
75.     $('#requestDiv').remove();
76.   }
77.   else { //Add Spare part - from Request
78.     $('#sparePart .select').remove();
79.     $('#newSparePart').remove();
80.     var sparePartsAddresses = SparePart.getSparepartAddress();
81.
82.     $.each(sparePartsAddresses, function (x, address) {
83.       sparePartsList.push(SparePart.getSparePartData1(address)[0]);
84.
85.       if (SparePart.getSparePartData1(address)[2] == true) {
86.         $('#sparePart').append('<option>' + SparePart.getSparePartData1(
87. address)[0] + '</option>');
88.       }
89.     });
90.     $('#button').click(function () {
91.       $('#loader').show();
92.       var rand = "0x1111111111111111111111111111111111111111".replace(/1/g, fu
93. nction () { return ((Math.random() * 16)).toString(16); });
94.       var addr = web3.personal.newAccount(rand);
95.       var sparePartRequestAddress = SparePartRequest.getSparePartRequestAddress
96. ([parameters.get('sprId')]);
97.       var status;
98.       if ($("#amount").val() > 0) {
99.         status = true;
100.      }
101.      else {
102.        status = false;
103.      }
104.
105.      if (parameters.get('sprId') == null) { //Add Spare part
106.        SparePart.setSparePartData1(addr, $("#name").val(), $("#descr
107. iption").val(), status, $("#amount").val(), { gas: 100000 }, (err, res) => {
108.          if (err) {
109.            $("#loader").hide();
110.            console.log(err);
111.          }
112.          window.location = "Index_Spare.html";
113.        });
114.      }
115.      else { //Add Spare part - from Request
116.        var id = sparePartsList.indexOf($("#sparePart").val())
117.        SparePartSparePartAddress = SparePart.getSparepartAddress()[id
118. ];
119.        var SparePartAmount = parseInt(SparePart.getAmount(SparePartA
120. ddress));
121.        var sparePartRequestAmount = parseInt($("#amount2").val());
122.
123.        if (SparePartAmount >= sparePartRequestAmount) {
124.          var newSparePartAmount = SparePartAmount - sparePartReque
125. stAmount;
126.          SparePart.setAmount(SparePartAddress, newSparePartAmount)
127. ;
128.          if ((parseInt(newSparePartAmount) == 0)) {
129.            SparePart.setStatus(SparePartAddress, false);
130.          }
131.          var ticketAddress = SparePartRequest.getSparePartRequestD
132. ata1(sparePartRequestAddress)[3];
133.          Ticket.setSparePart(ticketAddress, SparePartAddress);
134.          SparePartRequest.setStatus(sparePartRequestAddress, false
135. );
136.        }
```

```

132.             window.location = "Index_Spare.html";
133.         }
134.         else {
135.             alert("not enough spare parts");
136.             $("#loader").hide();
137.         }
138.     }
139. });
140. </script>
141. </body>
142.
143. </html>

```

7.2.2 Viewing a Spare Part

```

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4.     <meta charset="UTF-8">
5.     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6.     <meta http-equiv="X-UA-Compatible" content="ie=edge">
7.     <title>Document</title>
8.
9.     <link rel="stylesheet" type="text/css" href="main.css">
10.
11.     <script src="./node_modules/web3/dist/web3.min.js"></script>
12.
13. </head>
14. <body>
15.     <div class="container">
16.         <h1 id="headline">Spare Part</h1>
17.         <p id="sparePart"></p>
18.         <hr>
19.
20.         <h2>Restock</h2>
21.         <label for="sparePartAmount" class="col-lg-2 control-
label">Amount of Spare parts</label>
22.         <input id="sparePartAmount" type="number">
23.
24.         <button id="button">Save</button>
25.         <hr>
26.
27.         <button onclick="window.location.href='Index_Spare.html'">Back</button>
28.     </div>
29.
30.     <script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
31.
32.     <script>
33.         if (typeof web3 !== 'undefined') {
34.             web3 = new Web3(web3.currentProvider);
35.         } else {
36.             // set the provider you want from Web3.providers
37.             web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"))
38.         };
39.         web3web3.eth.defaultAccount = web3.eth.accounts[0];
40.
41.         var TicketContract = web3.eth.contract(...ABI...);
42.         var MachineContract = web3.eth.contract(...ABI...);
43.         var SparePartContract = web3.eth.contract(...ABI...);
44.         var SparePartRequestContract = web3.eth.contract(...ABI...);
45.
46.         var Ticket = TicketContract.at('0x7e06ccb5c46db5ed809d9d8f73132608495f894c')
;

```

```

47.     var Machine = MachineContract.at('0x7d51587098e47cbc2b3467e2e1eb3ab4a49650ad
    ');
48.     var SparePart = SparePartContract.at('0x82e4fb55e67dd09caa29ea1a7dc283180c13
    6396');
49.     var SparePartRequest = SparePartRequestContract.at('0x27c51c3ba389002c63b271
    5dfc7587aa04eb459c');
50.
51.     var parameters = new URLSearchParams(window.location.search);
52.     var sparePartAddress = SparePart.getSparepartAddress()[parameters.get('spId')
    ];
53.
54.     $( document ).ready(function() {
55.         $("#headline").append(" " + parseInt((parameters.get('spId'))+1));
56.
57.         var sparePartData1 = SparePart.getSparePartData1(sparePartAddress);
58.         var status;
59.
60.         if( sparePartData1[2] == true){
61.             status = "Available";
62.         }
63.         else{
64.             status = "Unavailable";
65.         }
66.         $("#sparePart").html('Name: ' + sparePartData1[0] +
67.             '<br> Description: ' + sparePartData1[1]
    +
68.             '<br> Status: ' + status +
69.             '<br> Amount: ' + sparePartData1[3]
70.             );
71.     });
72.
73.     $("#button").click(function() {
74.         console.log("address: "+sparePartAddress);
75.         var amount = parseInt($("#sparePartAmount").val());
76.         var SparePartAmount = parseInt(SparePart.getAmount(sparePartAddress));
77.
78.         if(amount >0){
79.             var x = parseInt(amount+SparePartAmount);
80.             SparePart.setStatus(sparePartAddress, true);
81.             SparePart.setAmount(sparePartAddress, x);
82.         }
83.         else{
84.             alert("Amount too low!");
85.             return;
86.         }
87.
88.         window.location="Index_Spare.html";
89.     });
90. </script>
91. </body>
92. </html>

```

7.3 Installation Guide

1) Step:

To run the prototype Node.js and NPM (Node Package Manager) is needed. You can check if you have them installed by typing the following commands in CMD.

```
>node -v
```

```
> npm -v
```

If the commands go unrecognized (or you have an older version than 10.16.0 for node and 6.9.0 for npm) please install/update them by downloading [Node.js](#).

Close the CMD and try the commands again, now they should work.

2) Step:

Now enter the following command:

```
> npm install -g ganache-cli
```

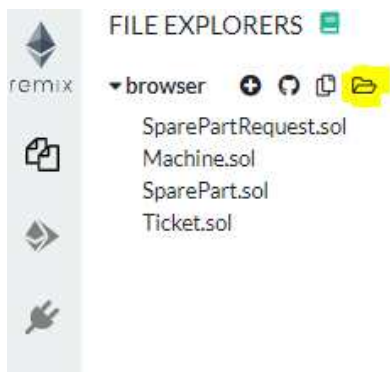
Now ganache is installed. Start it with the following command:

```
> ganache-cli
```

Now the blockchain is simulated and 10 different user accounts were created.

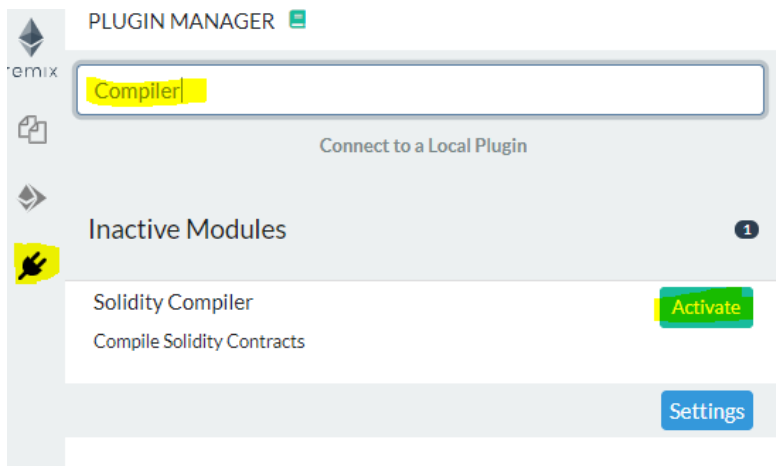
3) Step:

Now open <https://remix.ethereum.org> in Google Chrome and add the Smart contracts from the provided folder (Smart Contract) by clicking on the marked symbol and marking all files.

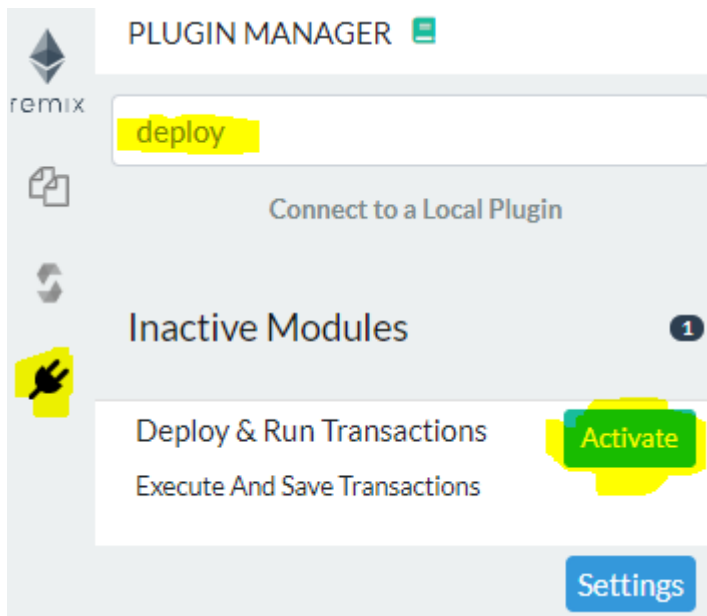


4) Step:


Click on the following symbol and search for compiler. Activate the module.

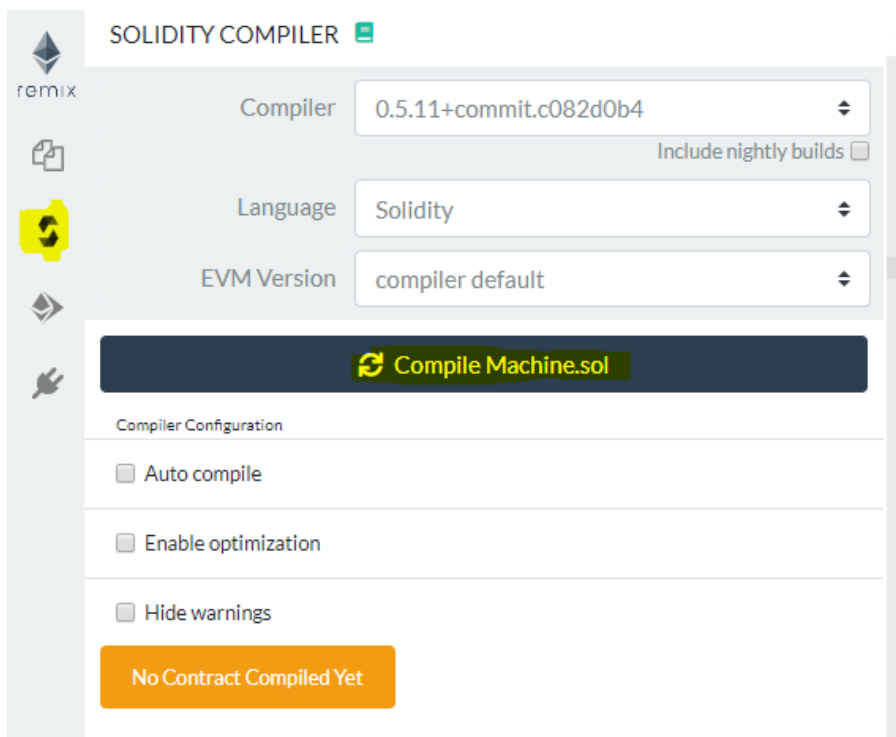


Click on the following symbol and search for deploy. Activate the module.



5) Step:

Now click on this symbol  and on the contract Machine.sol. Click on the following symbol on the left and press compile.



Go to the left to the following symbol and change the environment to Web3Provider. Accept the following popups. Then press “Deploy”. This will deploy the smart contract on the blockchain.

DEPLOY & RUN TRANSACTIONS

Environment: Web3 Provider

Account: Web3 Provider

Gas limit: 3000000

Value: 0 wei

MachineContract - browser/Machine.sol

Deploy

or

At Address Load contract from Address

Do this for all 4 contracts. The result should look like this:

DEPLOY & RUN TRANSACTIONS

Environment: Web3 Provider

Account: 0x615...1ee6a (99.9999999999992528018 ether)

Gas limit: 3000000

Value: 0 wei

TicketContract - browser/Ticket.sol

Deploy

or

At Address Load contract from Address

Transactions recorded: 4

Deployed Contracts

- MachineContract at 0x247...B461e (blockchain)
- SparePartRequestContract at 0x881...31Cb4 (blockchain)
- SparePartContract at 0x91a...CbeEB (blockchain)
- TicketContract at 0x323...8D1c0 (blockchain)

6) Step:

Go to the HTML folder and open the Contract.js file (with an editor). Change the contract addresses in Line 15-18 by copying them from Remix (by clicking on the symbol in the screenshot above). Save the file.

```
JS Contract.js X
C: > Users > glieb > Dropbox > Diplomarbeit > Blockchain > Programming > Demonstration > HTML > JS Contract.js > ...
1 |
2 | if (typeof web3 !== 'undefined') {
3 |   web3 = new Web3(web3.currentProvider);
4 | } else {
5 |   // set the provider you want from Web3.providers
6 |   web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));
7 | }
8 | web3.eth.defaultAccount = web3.eth.accounts[0];
9 |
10 | var TicketContract = web3.eth.contract([ { "constant": true, "inputs": [], "name": "countTickets",
11 | var MachineContract = web3.eth.contract([ { "constant": false, "inputs": [ { "name": "_machineNr",
12 | var SparePartContract = web3.eth.contract([ { "constant": false, "inputs": [ { "name": "_sparePartN
13 | var SparePartRequestContract = web3.eth.contract([ { "constant": true, "inputs": [], "name": "count
14 |
15 | var Ticket = TicketContract.at('0xc481544baed86dacdd70bfe565d27956039d7640');
16 | var Machine = MachineContract.at('0x79aaeba85628a444a423e14f3df792422c49938a');
17 | var SparePart = SparePartContract.at('0xe201df9a19d6d5a05ee1142f0b38a2ab25e8e973');
18 | var SparePartRequest = SparePartRequestContract.at('0x9cdc9438495ca3444cbf1ae5c7c7230840d39670');
19 |
20 |
```

7) Step:

Now the front-end application is connected to the blockchain. Just open the HTML files “Index” and “Index_Spare” in Google Chrome and you can start testing.

8 List of figures

Figure 1 – Centralized server client model (left) & Decentralized peer to peer model (right) (Park et al. 2018, p. 10).....	5
Figure 2 – Research Framework (Hevner et al. 2004).....	8
Figure 3 - Structure of the thesis	10
Figure 4 - Industrial revolutions (Xu et al. 2018, p. 2943).....	12
Figure 5 - View of a Cyber-physical system (Yao et al. 2017, p. 3)	14
Figure 6 – Applications of the 5 level architecture for CPS (Lee et al. 2015, p. 20)..	15
Figure 7 - Automation hierarchy and distributed services (Monostori et al. 2016, p. 626)	16
Figure 8 - CPS based manufacturing architecture – (Yao et al. 2017, p. 6)	16
Figure 9 - Limits of manufacturers (Allwood et al. 2016, p. 730).....	20
Figure 10 - Blockchain adaption (Hughes, Alex et al. 2019, p. 6)	23
Figure 11 – Blockchain example (Nofer et al. 2017, p. 184).....	24
Figure 12 - Linking of Advantages, Disadvantages, Risks and Challenges.....	41
Figure 13 - Cloud-Based Design and Manufacturing (Thames and Schaefer 2017, p. 12)	45
Figure 14 - Distributed infrastructure with centralized interfacing system (Thames and Schaefer 2017, p. 19)	46
Figure 15 - Software defined cloud manufacturing architecture (Thames and Schaefer 2017, p. 26)	47
Figure 16 - Example of collaboration on a CAD model (Thames and Schaefer 2017, p. 36)	48
Figure 17 - Manufacturing workflow (Thames and Schaefer 2017, p. 63)	51
Figure 18 - SCADA architecture (Thames and Schaefer 2017, p. 78).....	54
Figure 19 - Communication zones of a SCADA system (Ahmed et al. 2012, p. 46)..	55
Figure 20 - Meta cloud data storage architecture, adapted from (Thames and Schaefer 2017, p. 115)	60
Figure 21 - IoT remote maintenance architecture (Thames and Schaefer 2017, p. 237)	61
Figure 22 - Secure modular IoT Unit (Thames and Schaefer 2017, p. 238).....	62
Figure 23 - Use-case blockchain network.....	64
Figure 24 - Theoretical model.....	65
Figure 25 - Meta Mask Chrome add-on.....	67
Figure 26 - Ganache client	67
Figure 27 - Deploying a contract in Remix.....	68
Figure 28 - User actions	68
Figure 29 - System architecture.....	73
Figure 30 - Data model.....	74
Figure 31 - Part of the ABI from the Machine contract in Remix.....	78

Figure 32 - Machine dashboard.....	80
Figure 33 - Machine detail view	81
Figure 34 - Ticket detail view	81
Figure 35 - Spare Part dashboard	82
Figure 36 - Spare Part detail.....	82
Figure 37 - Spare Part Request.....	83
Figure 38 - Fulfil Request	83
Figure 39 - Comparison of IT affinity and time on each task	87
Figure 40 – Concept for a remote maintenance architecture using blockchain	92

9 List of tables

Table 1 Guidelines (Hevner et al. 2004)	9
Table 2 - Design principles for Industry 4.0 (Hermann et al., p. 11).....	17
Table 3 – Summary of challenges of smart manufacturing.....	19
Table 4 - Advantages and disadvantages of smart manufacturing.....	21
Table 5 - Existing blockchain prototypes (Lamberti et al. 2017, p. 5–6) and (Gatteschi et al. 2018, p. 6).....	23
Table 6 - Blockchain definitions	25
Table 7 - Consensus algorithms (Lang et al. 2018, p. 1096)	26
Table 8 - Blockchain implementations	27
Table 9 - Blockchain challenges	29
Table 10 - Limitations of blockchain technology	29
Table 11 - Advantages of blockchain.....	30
Table 12 - Disadvantages of blockchain	32
Table 13 - Mapping challenges of smart manufacturing to blockchain benefits.....	34
Table 14 - SWOT analysis: adopting blockchain (Gatteschi et al. 2018, p. 9–11)	35
Table 15 - Mapping of blockchain advantages to Smart manufacturing demands.....	38
Table 16 - Mapping of blockchain disadvantages to Smart manufacturing demands	39
Table 17 - Technological and non-technological risks	41
Table 18 - Use-cases and possible application areas in smart manufacturing	43
Table 19 - Security for collaboration (Thames and Schaefer 2017, p. 38) and (Wu et al. 2016, p. 466).....	49
Table 20 - Security concepts (Moon et al. 2018, p. 6)	50
Table 21 - Focus areas of Security (Thames and Schaefer 2017, p. 192)	53
Table 22 - IoT layers (Thames and Schaefer 2017, p. 111)	57
Table 23 - Security vulnerabilities and solutions of IoT (Thames and Schaefer 2017, p. 113)	58
Table 24 - Potential application sectors and the impact	63
Table 25 - Description of use-case Create Machine.....	69
Table 26 - Description of use-case View Machine.....	69
Table 27 - Description of use-case Update status or comment	70
Table 28 - Description of use-case Create Ticket.....	70
Table 29 - Description of use-case View Ticket.....	70
Table 30 - Description of use-case Create Spare Part Request.....	71
Table 31 - Description of use-case Create Spare Part	71
Table 32 - Description of use-case Update Spare Part	71
Table 33 - Description of use-case Update Spare Part Request	72
Table 34 - Non-functional requirements	72
Table 35 - Overview of the metric results	88

10 List of Code Fragments

Listing 1 - Smart contract for Spare parts	76
Listing 2 - Section of front-end code	77
Listing 3 - Section of HTML file for creating a Spare Part.....	78
Listing 4 - Section of HTML file for viewing a Spare part	79

11 List of abbreviations

\$	Dollar
&	and
3D	Three-dimensional
ABI	Application Binary Interface
API	Application programming interface
CaaS	Components-as-a-Service
CAD	Computer-Aided Design
CBDM	Cloud-Based Design and Manufacturing
CPPS	Cyber-physical production systems
CPS	Cyber Physical Systems
ERP	Enterprise Resource Planning
GB	Gigabyte
GUI	Graphical user interface
HMI	Human Machine Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
i.e.	Id est (=that is)
IaaS	Infrastructure-as-a-Service
ICT	Information and Communication Technology
IoS	Internet of Services
IoT	Internet of Things
IS	Information System
MES	Manufacturing Execution System
MQTT	Message Queuing Telemetry Transport
P2P	Peer to peer
PaaS	Platform-as-a-Service
PLC	Programmable Logical Controller
QC	Quality Control
RTU	Remote Terminal Unit
SaaS	Software-as-a-Service
SCADA	Supervisory Control and Data Acquisition
SM	Smart Manufacturing