

Development of a framework for tracking goods in manufacturing networks using a distributed ledger technology

by
Fabian Tobias Dietrich

*Thesis presented in fulfilment of the requirements for the degree of
Master of Engineering (Engineering Management)
in the Faculty Engineering at Stellenbosch University
This thesis has also been presented at Reutlingen University, Germany, in terms
of a double-degree agreement.*

Supervisors: Dr. Louis Louw
Co-supervisor: Dr. Daniel Palm

March 2020

Declaration

By submitting this project electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

March 2020

Copyright © 2020 Stellenbosch University

All rights reserved

Abstract

Globalisation, shorter product life cycles, and increasing product varieties have led to complex supply chains. At the same time, there is a growing interest by customers and governments in having a greater transparency of brands, manufacturers, and producers throughout the supply chain. Due to the complex structure of collaborative manufacturing networks, the increase in supply chain transparency is a challenge for manufacturing companies. Distributed ledger technologies offer an innovative solution to increase the transparency, security, authenticity, and auditability of products. However, there are still uncertainties when applying the distributed ledger technology to manufacturing scenarios and thus enable all stakeholders to trace the audit history of each component of an assembled product. This research work proposes a framework to increase the transparency and auditability of products in collaborative manufacturing networks by adopting the distributed ledger technology. The framework considers the challenges of manufacturing supply chains with the opportunities of distributed ledger technologies and combines them in a conceptual implementation process of supply chain management systems. In this context, each component of a product is represented by a unique virtual identity generated by distributed ledger-based smart contracts. These virtual identities can only be sent and merged if defined conditions specified in smart contracts are met. This enables all physical processes and their dependencies to be mapped in the distributed ledger. The results, based on a practical implementation of the framework, show that a transparent auditability of assembled products and all the components they consist of can be achieved. Applications based on the proposed framework can currently only enable real-time tracking reliably in permissioned networks. The implementation on a permissionless network provides full transparency for all stakeholders including the customer, while the implementation on a permissioned network only provides a restricted transparency for the customers.

Keywords: Manufacturing Supply Chain; Distributed Ledger Technology; Tracking; Transparency

Opsomming

Globalisering, korter produklewensiklusse en toenemende variëteite het gelei na komplekse voorsieningskettings. Terselfdertyd is daar 'n toenemende belangstelling van kliënte en regerings vir groter deursigtigheid van handelsmerke, vervaardigers en produsente in die hele verskaffingsketting. As gevolg van die ingewikkelde struktuur van samewerkende vervaardigingsnetwerke, is die toename van voorsieningsketting deursigtigheid 'n uitdaging vir vervaardigingsondernemings. Verspreide grootboek tegnologieë bied 'n innoverende oplossing om die deursigtigheid, sekuriteit, egtheid en ouditeerbaarheid van produkte te verhoog. Daar is egter steeds onduidelikhede oor die toepassing van die verspreide grootboektegnologie op vervaardiging prosesse en stel alle betrokke partye in staat om die ouditgeskiedenis van elke komponent van 'n saamgestelde produk op te spoor. Hierdie navorsing stel 'n raamwerk voor wat die deursigtigheid en ouditeerbaarheid van produkte in samewerkende vervaardigingsnetwerke te verhoog deur die verspreide grootboektegnologie te gebruik. Die raamwerk oorweeg die uitdagings van die vervaardiging van voorsieningskettings met die geleentheid van verspreide grootboektegnologieë en verbind dit in 'n konseptuele implementeringsproses van voorsieningskettingbestuurstelsels. In hierdie konteks word elke komponent van 'n produk voorgestel deur 'n unieke virtuele identiteit wat gegenereer word deur verspreide grootboek-gebaseerde “smart contracts”. Hierdie virtuele identiteite kan slegs gestuur en saamgevoeg word as die voorwaardes wat in “smart contracts” gespesifiseer is, nagekom word. Dit stel alle fisiese prosesse en hul afhanklikhede in die verspreide grootboek in staat om gealokeer te word. Die resultate, gebaseer op 'n praktiese implementering van die raamwerk, toon aan dat 'n deursigtige auditering van saamgestelde produkte en al die komponente waaruit hulle bestaan, bereik kan word. Aansoeke wat op die voorgestelde raamwerk gebaseer is, kan tans slegs “real time-tracking” betroubaarheid verseker in toestemmingsnetwerke. Die implementering op 'n toestemmingslose netwerk bied volledige deursigtigheid vir alle betrokke partye, insluitende die kliënt, terwyl die implementering op 'n toegelate netwerk slegs 'n beperkte deursigtigheid vir die kliënte bied.

Sleutelwoorde: Vervaardiging verskaffingsketting; Verspreide Grootboektegnologie; Volging; Deursigtigheid

Acknowledgements

I wish to express my sincere appreciation to my supervisors, Dr. Daniel Palm and Dr. Louis Louw. Without their persistent help, guidance, and understanding, the goal of this research project would not have been realised. I would also like to thank all employees of the ESB Werk150 for their valuable suggestions and their willingness to help throughout this research work. Their assistance was a milestone in the practical completion of this project.

Additionally, I wish to acknowledge the support and great love of my siblings - my sisters, Stefanie and Miriam; and my brothers, Yannik and Lukas. You all provide me with unending joy and inspiration. Furthermore, I would like to express my special gratitude to my grandpa Hans. Thank you for every hour we have spent together and for being such a wonderful English teacher. Special thanks also to my girlfriend Petra for always being patient, empathetic, and honest. My deepest appreciation belongs to my parents. Without their unconditional love, care, and support, I would have never made it this far. I appreciate this more than you will ever know.

The Author

November 2019

Table of contents

Declaration	II
Abstract	III
Opsomming	IV
Acknowledgements	V
Table of contents	VI
List of figures	IX
List of tables	XII
List of listings	XIII
List of acronyms	XIV
1 Introduction	1
1.1 Background and rationale of the research	3
1.2 Research problem statement and questions	5
1.3 Research objectives and contribution	6
1.4 Research methodology and design overview	7
1.5 Dissertation outline	9
2 Literature Review	10
2.1 Fundamentals: manufacturing supply chain	10
2.1.1 General structure of manufacturing supply chains	10
2.1.2 Complexity drivers of manufacturing supply chains	12
2.1.3 Supply chain transparency	14
2.1.4 Supply chain management systems	15
2.1.4.1 <i>Supply chain design</i>	17
2.1.4.2 <i>Supply chain planning</i>	17
2.1.4.3 <i>Supply chain execution</i>	18
2.1.4.4 <i>Tracking and tracing systems to increase transparency</i>	19
2.2 Fundamentals: Distributed ledger technology	21

2.2.1	From centralised to decentralised systems	21
2.2.2	Blockchain technology	26
2.2.2.1	<i>Hashing in BCTs</i>	29
2.2.2.2	<i>Transaction logic</i>	30
2.2.2.3	<i>Consensus algorithm</i>	33
2.2.2.4	<i>Smart contracts</i>	39
2.3	Existing use cases combining distributed ledger technologies and supply chains	43
2.3.1	Blockchain technology for tracking goods	43
2.3.2	Blockchain technology for tracking carriers	44
3	Research environment	46
3.1	The manufacturing network of Werk150	46
3.2	Country-specific circumstances.....	48
4	Framework development	51
4.1	Network design stage.....	51
4.1.1	Identification of all stakeholders.....	52
4.1.2	Assignment of network access authorisation	53
4.2	Network planning stage	55
4.2.1	Definition of product composition.....	56
4.2.1.1	<i>Determination of virtual identities</i>	59
4.2.2	Definition of a smart contract logic	61
4.2.2.1	<i>Prediction of transactions per second</i>	65
4.2.3	Platform decision	67
4.2.3.1	<i>Permissionless or permissioned platform</i>	69
4.3	Execution stage.....	76
4.3.1	DApp implementation.....	76
4.3.2	Shop floor implementation	80
4.4	Proposed framework.....	81
5	Practical implementation and dApp development	85
5.1	Implementation of the network design stage	85
5.2	Implementation of the network planning stage	86
5.2.1	Implementation of the platform decision	91
5.2.1.1	<i>Methodology of the platform selection</i>	92
5.2.2	Set up of a permissioned Ethereum network	99
5.3	Implementation of the network execution stage.....	100

5.3.1	Development of the dApp.....	100
5.3.2	Operational process description.....	104
6	Verification and validation of the framework.....	108
6.1	Smart contract immutability and security.....	108
6.2	Tracking manufactured goods by using a distributed ledger technology	110
6.2.1	Time accuracy differences between a permissioned and permissionless application	112
6.3	Preventing counterfeit parts entering the supply chain.....	114
6.4	Customer transparency	115
7	Summary, conclusions, and recommendations.....	118
7.1	Research summary.....	118
7.2	Research conclusion and contribution.....	120
7.3	Limitations and recommendations for further research.....	125
	References	127
	Appendix A: Smart contract	140
	A1: Smart contract Solidity source code	140
	A2: Smart contract ABI source code.....	144
	Appendix B: Graphical user interfaces.....	153
	B1: User interface source code.....	153
	B2: User interface layout.....	157
	B3: Certifier interface layout.....	158
	B4: Supplier interface layout.....	159
	B5: Manufacturer interface layout.....	160

List of figures

Figure 1: Aspects of digitalisation that can drive the most value	2
Figure 2: Hype cycle for emerging technologies	3
Figure 3: Methodology and design of the research.....	8
Figure 4: Structure of the thesis	9
Figure 5: Network structure of a typical manufacturing supply chain	11
Figure 6: a.) Non-modular assembly; b.) Modular assembly	12
Figure 7: Task model of supply chain management software systems.....	16
Figure 8: A physically distributed system with centralised control.....	21
Figure 9: Typical structure of a centralised system	22
Figure 10: Structure of a decentralised system	23
Figure 11: A decentralised peer-to-peer system	23
Figure 12: a.) Distributed ledger technology arrangement with nodes hosted within a single entity; b.) Distributed ledger technology arrangement with multiple entities, each hosting a node.....	25
Figure 13: Linked blocks by reference to the previous block header hash.....	28
Figure 14: Workflow of blockchain transactions.....	32
Figure 15: Transactions (TX) hashed in a merkle-tree	33
Figure 16: Structure of a decentralised application	41
Figure 17: Mechanism of Ethereum smart contracts	42
Figure 18: a) Main parts of the handlebar stem; b) Main parts of the substructure.....	46
Figure 19: Assembly Structure	47
Figure 20: The world's most respected 'Made in' labels	50
Figure 21: Considered steps during the network design stage.....	51
Figure 22: Peer-to-peer architecture using a supernode	54

Figure 23: Considered steps during the network planning stage	55
Figure 24: Relationship of all entities directly involved in the manufacturing process.	56
Figure 25: Distributed ledger technologies as verified public timestamps.....	57
Figure 26: Model to create virtual identities for assets without dependencies	62
Figure 27: Model to create virtual identities for assets with dependencies	63
Figure 28: Model to send virtual identities of assets	64
Figure 29: Smart contract models arranged in an alternating sequence	65
Figure 30: Correlation between manufacturing processes and total number of transactions ..	66
Figure 31: a.) Tracking process with each entity owing one address; b.) Detailed tracking process where Entity A owns many addresses	66
Figure 32: Part 1 of the platform decision flow	69
Figure 33: Adoption of foundational technologies	70
Figure 34: The scalability trilemma	71
Figure 35: Energy consumption by country including Ethereum	72
Figure 36: Simplified proof-of-work in a permissioned network with a fixed difficulty and a fixed number of participants	73
Figure 37: Platform decision flow	75
Figure 38: Considered steps during the execution stage.....	76
Figure 39: Permissionless platform implementation scheme	78
Figure 40: Permissioned platform implementation scheme.....	79
Figure 41: Complexity drivers influencing the network design stage	82
Figure 42: Complexity drivers influencing the network planning stage.....	83
Figure 43: Conceptual framework for tracking goods in manufacturing networks using a distributed ledger technology.....	84
Figure 44: Dependencies of all included assets	87

Figure 45: Holistic smart contract logic models of the manufacturing supply chain89

Figure 46: Process sequence of the certificate creation104

Figure 47: Process sequence of the component production.....105

Figure 48: Process sequence of the assembly manufacturing.....106

Figure 49: User interface to retrieve all information about the composition and the history of an assembly.....107

Figure 50: Attempting to access a functions with an unauthorised address results in an indispensable error of the system.....109

Figure 51: Database structure linking the virtual identities in a course of time111

Figure 52: Transaction confirmation times of the Ethereum Rinkeby network112

Figure 53: User interface layout157

Figure 54: Certifier interface layout158

Figure 55: Supplier interface layout159

Figure 56: Manufacturer interface layout160

List of tables

Table 1: Primary research question.....	5
Table 2: Secondary research questions	6
Table 3: Main complexity drivers of manufacturing supply chains	14
Table 4: Comparisons among public, consortium, and private distributed ledgers.....	26
Table 5: Minor changes of the input result in totally different hashes	29
Table 6: Characteristics of public and permissioned smart contracts.....	40
Table 7: Comparison between primary keys in relational central databases and Hash IDs	60
Table 8: Scalability comparison between DLT platforms	71
Table 9: Enlistment of all stakeholders, their roles, and their network authorisations.....	86
Table 10: Assets with their respective shared information.....	88
Table 11: Sources used for the platform selection.....	93
Table 12: Basic parameter comparison between the platforms Ethereum, EOS, and Hyperledger Fabrics.....	94
Table 13: Rating scale used for utility analysis	96
Table 14: Result of the utility analysis in order to select a suitable platform.....	97
Table 15: Roles and their corresponding Ethereum addresses	100
Table 16: Primary research question 1.....	120
Table 17: Secondary research question 1.....	121
Table 18: Secondary research question 2.....	122
Table 19: Secondary research question 3.....	122
Table 20: Secondary research question 4.....	123
Table 21: Secondary research question 5.....	124

List of listings

Listing 1: Source code of the genesis file	99
Listing 2: Source code for defining the roles at the moment of smart contract deployment.	101
Listing 3: Connection between memory and ‘get’ function in the source code	101
Listing 4: Source code of the assembly’s products composition.....	102
Listing 5: Simplified source code for the creation of the Hash ID of Component A	102
Listing 6: Source code to receive the history of the assembly.....	103
Listing 7: JSON ABI resulting from the history function of the assembly	103
Listing 8: Solidity source code of the smart contract	143
Listing 11: Smart contract ABI source code.....	152
Listing 12: React user interface source code	156

List of acronyms

ABI	Application Binary Interface
AEI	Automatic Equipment Identification
AI	Artificial Intelligence
ASIC	Application-specific Integrated Circuit
BCT	Blockchain Technology
dApp	Decentralised Application
DBFT	Delegated Byzantine Fault Tolerance
DLT	Distributed Ledger Technology
ERP	Enterprise Resource Planning
EVM	Ethereum Virtual Machine
GPS	Global Positioning System
GUI	Graphical User Interface
Hash ID	Decentral Identification Number Created by Hashing Algorithms
ICO	Initial Coin Offering
ID	Identification Number
IoT	Internet of Things
IPO	Initial Public Offering
NGO	Non-governmental Organisation
NSA	National Security Agency
OEM	Original Equipment Manufacturer
P2P	Peer-to-Peer
PaaS	Platform-as-a-Service
PBFT	Practical Byzantine Fault Tolerance
PoS	Proof-of-Stake
PoW	Proof-of-Work
QR Codes	Quick Response Codes
SC	Supply Chain
RFID	Radio Frequency Identification
SCM	Supply Chain Management
SCV	Supply Chain Visibility
SHA-256	Secure Hash Algorithm with a 256-bit length

SUP	Suspected Unapproved Part
TPS	Transactions per second
VI	Virtual Identity

1 Introduction

Globalisation has an impact on every country regardless of its economic, political or social situation and it is not slowing down. In fact, globalisation has initiated an innovation-driven era that is mainly characterised by intense competition, shorter product life cycles, and high product varieties (Ernst & Haar, 2019). The rapid changes due to globalisation and the increasing complexity of supply chains also influence the manufacturing landscape (Mourtzis & Doukas, 2012; Ueda, Takenaka, Vancza, & Monostori, 2009). Thus, properly configured and easily adaptable manufacturing networks are required, which are capable of handling the complexity and magnitude of the supply chain structures. These qualities represent a critical factor for companies in order to maintain their viability (Mourtzis, Doukas, & Psarommatis, 2015, p. 274).

In addition to the structural complexity of global supply chains, companies have to deal with the growing interest of customers, governments, and non-governmental organisations (NGOs) in having a greater transparency of brands, manufacturers, and producers throughout the supply chain. For manufacturers, social and environmental sustainability issues have become increasingly important in order to maintain a flawless brand reputation. However, as supply chains become more global, many suppliers in the network may be located in developing countries where governments have only a limited ability and willingness to enforce their own laws. Therefore, the distributed nature of today's supply chains creates increasing levels of risk for multinational businesses, making transparency of supply chains both critical and complex (Chen, Zhang, & Zhou, 2018; Linich, 2014; New, 2010). Especially for organisations operating in complex and dispersed supply chains, the expansion of measures to increase supply chain visibility can be of great advantage in reducing these risks (Brandon-Jones, Squire, Autry, & Pertersen, 2014; Linich, 2014).

Figure 1 shows the results of a survey conducted by SCDigest and JDA in 2016 to point out which aspects of digitalisation can drive the most value. According to 203 industry experts, supply chain visibility with a rating of 59% is the aspect that drives the most value. This emphasises the high importance of visibility for organisations and their supply chains.

Which aspects of digitalisation that can drive the most value?

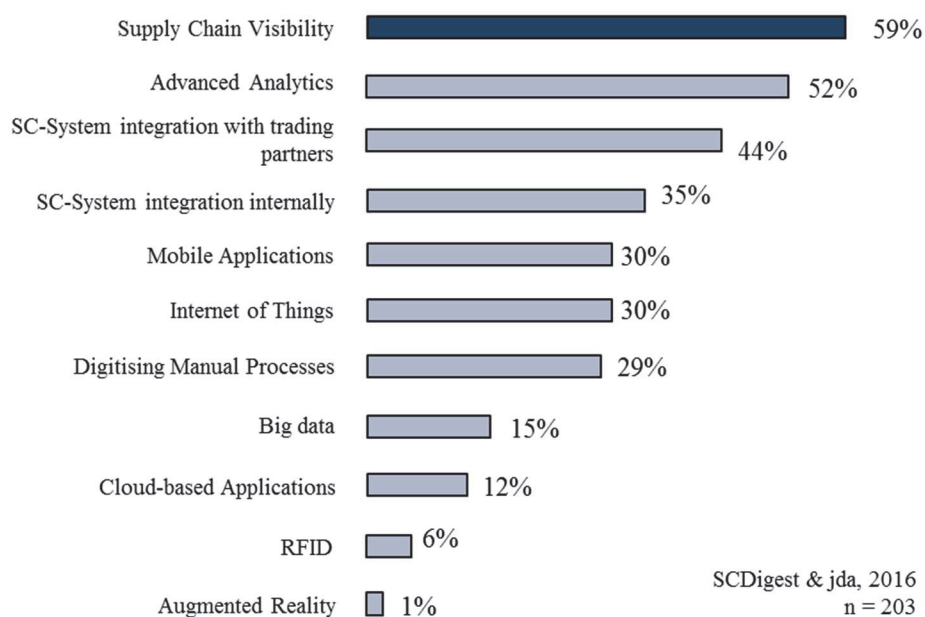


Figure 1: Aspects of digitalisation that can drive the most value (SCDigest & jda, 2016)

Figure 1 shows the results of a survey conducted by SCDigest and JDA in 2016 to point out which aspects of digitalisation can drive the most value. According to 203 industry experts, supply chain visibility with a rating of 59% is the aspect that drives the most value. This emphasises the high importance of visibility for organisations and their supply chains.

In October 2008, the pseudonym Satoshi Nakamoto published the famous Bitcoin white paper and thus described the blockchain technology for the first time. By the help of this technology and Bitcoin as a cryptocurrency, the paper clearly aims to change the traditional financial sector and to make trusted third parties superfluous (Nakamoto, 2008). As a result of bitcoin's success in the year 2009, the blockchain technology was mainly associated with financial applications at this time (Grinberg, 2011; Kaplanov, 2012; Sorge & Krohn-Grimberghe, 2012). In 2013, Vitalik Buterin extended the idea behind Bitcoin and introduced the white paper of Ethereum. Compared to Bitcoin, the Ethereum protocol moves far beyond using the blockchain technology just as a currency. It is a decentralised platform that allows users to theoretically create any type of application (Buterin, 2013). The possibility to run decentralised applications on blockchain platforms initiated a hype around the technology with inflated expectations. Figure 2 shows the hype cycle for emerging technologies according to Gartner (2018).

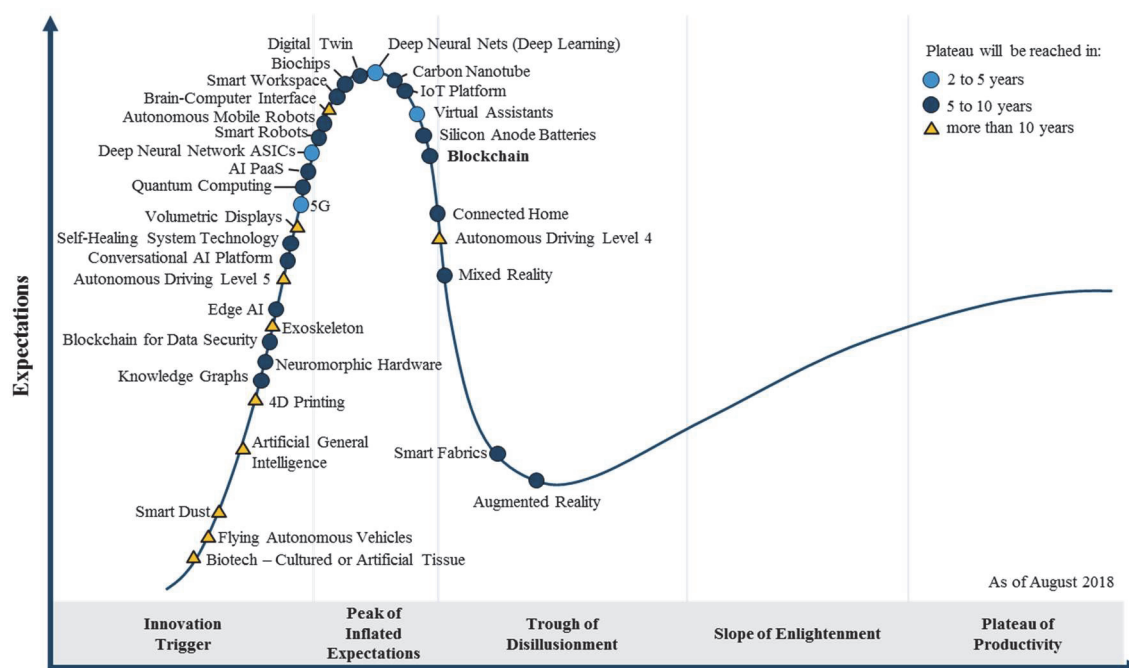


Figure 2: Hype cycle for emerging technologies (based on Gartner, 2018)

Gartner's yearly published hype cycles exemplify the development of the blockchain technology from cryptocurrencies and financial applications to decentralised application platforms with several fields of application. In 2015, Gartner specifically listed cryptocurrencies and arranged them sliding right into the trough of disillusionment phase (Gartner, 2015). In the hype cycle of 2016, however, the term cryptocurrencies was replaced by the term blockchain, which was done even before the peak of inflated expectations. The new potential of the blockchain technology to become a decentralised application platform has been elucidated through this expansion (Gartner, 2016). Especially when applying the blockchain technology to supply chain management, companies have high expectations to solve the transparency and auditability issues of complex collaborative supply chains (Gentemann, 2019; Iansiti & Lakhani, 2017; Panetta, 2018).

1.1 Background and rationale of the research

A lack of supply chain transparency can result in various vulnerabilities of manufacturing supply chains. For example, in the automotive industry, counterfeit parts are increasingly putting consumer's safety at risk. Automotive parts that are frequently counterfeited in huge volumes are, for example, airbags, engine and drivetrain components, brake pads, automotive body parts, electrical components, wheels, and windscreens (Peresson, 2019). In particular,

the counterfeiting of electronic parts causes potential risks including safety and loss of profits to companies, as well as maligning the reputation of manufacturers and distributors. Due to the complexity of global supply chains, it becomes increasingly difficult to maintain the traceability of all components even from very reputable authorised distributors. This increases the probability of counterfeit components being introduced into a supply chain (Collier, Hassler, Lambert, DiMase, & Linkov, 2019; DiMase, Collier, Carlson, Gray, & Linkov, 2016; Pecht, 2013).

Besides the complexity of supply chains, the storage of component-related data in central systems or the existence of paper-based certificates also increases the probability of counterfeit parts entering the supply chain. For example, in the aviation industry, organisations file paper documentation of parts such as certificates of conformance, packing lists, and test documentation in archives and store the data in central systems according to their own records information policy. Typically, the certificates are not directly linked with the physical batch/shipment of parts. After a certain record retention period has elapsed, organisations are allowed to destroy the paperwork, while the physical product may still be in circulation (DiMase et al., 2016). This results in a confusing number of suspected unapproved parts (SUBs), which are aircraft parts that do not qualify to meet the provisions of an approved part and do not meet the quality constraints of the industry. Thus, SUBs are seriously violating the strict aircraft security standards. (Acharjya & Geetha, 2017, p. 263)

An important method to avoid the incorporation of tampered counterfeit parts in assemblies is to gain complete traceability of all parts and therefore to increase the transparency throughout the whole supply chain. However, achieving full transparency and detecting counterfeit components is extremely complex and can be a costly undertaking. (Collier et al., 2019; Guin et al., 2014; Machado, Paiva, & da Silva, 2018). In addition, this turns out to be even more challenging, since counterfeit parts often originate in developing countries where governments have only limited abilities to enforce laws (Chen et al., 2018; Domon, 2018).

To improve avoidance and detection of overproduced, cloned, and tampered counterfeit types represents a present research gap. Specifically, the problem of incorporating

tampered counterfeit parts in assemblies introduces a vulnerability that must be prevented (Collier et al., 2019, p. 450).

1.2 Research problem statement and questions

The purpose of this work is to investigate how manufacturing networks can increase the transparency for all stakeholders. In that respect, this work focuses on the problem of complex manufacturing supply chains creating products consisting of several components. The incorporating of parts with different origins into assembly lines complicates the auditability of each part after the assembly process. This leads to vulnerabilities in manufacturing networks allowing counterfeit parts to enter the supply chain. These vulnerabilities not only jeopardise the reputation of manufacturers because of products consisting of counterfeit parts, but also because of the increased risk of components being unwittingly produced under reprehensible social or environmental circumstances. Distributed ledger technologies such as the blockchain technology are frequently associated as promising technologies to increase supply chain transparency. The primary research question (Table 1), aims to investigate how distributed ledger technologies can be adopted to solve the auditability problems of manufacturing networks.

Table 1: Primary research question

Primary research question 1	How can manufacturing networks adopt the distributed ledger technology to enable a transparent tracking of assembled goods and the auditability of all components they consist of?
------------------------------------	--

The following secondary research questions shown in Table 2 must be considered in order to answer the primary research questions adequately. Answering the secondary research questions leads to findings that serve to clarify the primary research questions.

Table 2: Secondary research questions

Secondary research question 1	How can the physical assets be linked to a virtual identity on the distributed ledger?
Secondary research question 2	How can the distributed ledger technology reduce vulnerabilities of counterfeit parts entering the supply chain?
Secondary research question 3	What requirements does a platform have to meet in order to be eligible for a practical implementation?
Secondary research question 4	How does an implementation differ from the use of a permissionless or permissioned network?
Secondary research question 5	What does a practical implementation of a distributed ledger technology for tracking goods in manufacturing supply chains look like?

1.3 Research objectives and contribution

In order to achieve the aim of the study, nine main objectives are developed. These objectives serve to guide the study in the intended direction and to keep the focus on the aim of this research work. The following objectives are defined to support the research:

- i. Critically review definitions of manufacturing supply chains and supply chain transparency (Section 2.1.1 + Section 2.1.3)
- ii. Examine critically the structures and existing literature of manufacturing supply chains (Section 2.1)
- iii. Examine critically the structure and existing literature of distributed ledger technologies (Section 2.2)
- iv. Investigate existing projects combining distributed ledger technologies and supply chains (Section 2.3)
- v. Analyse the relevance of the topic in terms of the country-specific circumstances of South Africa and Germany (Section 3.2)

- vi. Develop a framework for manufacturing networks to increase the supply chain transparency by adopting distributed ledger technologies (Chapter 4)
- vii. Develop a solution to create unique smart contract-based virtual identities (Section 4.2.1)
- viii. Develop a decentralised application based on the outcome of the framework and implement it in the research environment (Chapter 5)
- ix. Verify and validate the feasibility of the proposed framework (Chapter 6)

1.4 Research methodology and design overview

The methodology for this thesis is the development of a conceptual framework. Jabareen (2009, p. 51) defines a conceptual framework as a “network, or ‘a plane’, of interlinked concepts that together provide a comprehensive understanding of a phenomenon or phenomena”. A conceptual framework is arranged in a logical structure to display how ideas in a study relate to one another (Grant & Osanloo, 2014). It is the researcher’s own constructed model to explain the relationship between the main variables in the study (Adom, Hussein, & Joe, 2018, p. 440). Jabareen (2009) suggests to build a conceptual framework from existing multidisciplinary literature since it is a process of theorisation, which “uses grounded theory methodology rather than a description of the data and the targeted phenomenon”. The aim of this research is to combine the requirements of manufacturing networks with the possibilities of distributed ledger technologies. Therefore, the respective areas are first examined in an extensive literature review. This literature review serves as a basis for the subsequent framework development. Leavy (2017) states that qualitative approaches, in particular, are suitable for exploring complex contexts in cases where only little literature is available. In this context, *inter alia*, experiments are important data collection options. Since no similar approach exists in literature, this research verifies and validates the feasibility of the framework based on a practical implementation in the manufacturing network of Reutlingen University and Stellenbosch University.

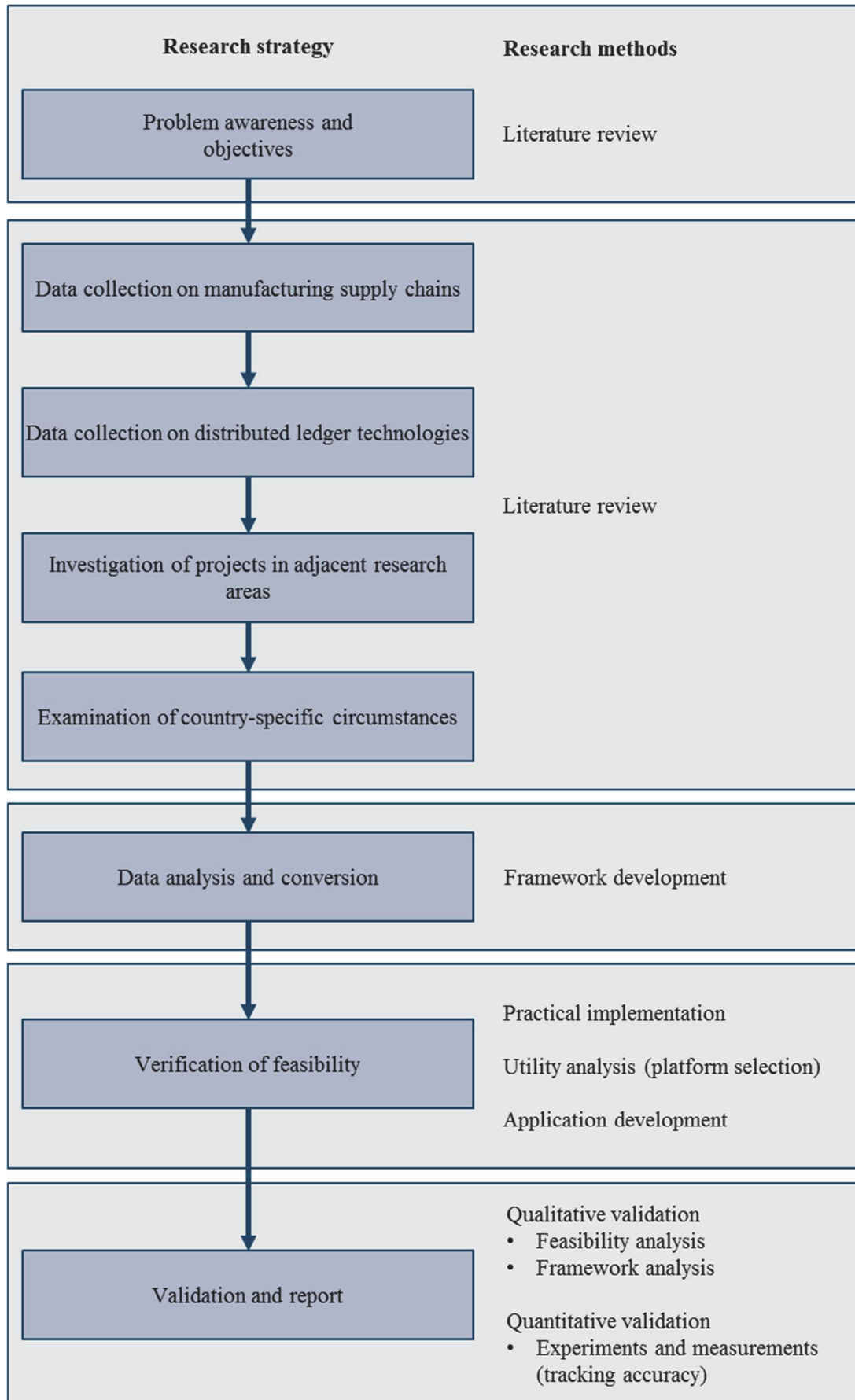


Figure 3: Methodology and design of the research

1.5 Dissertation outline

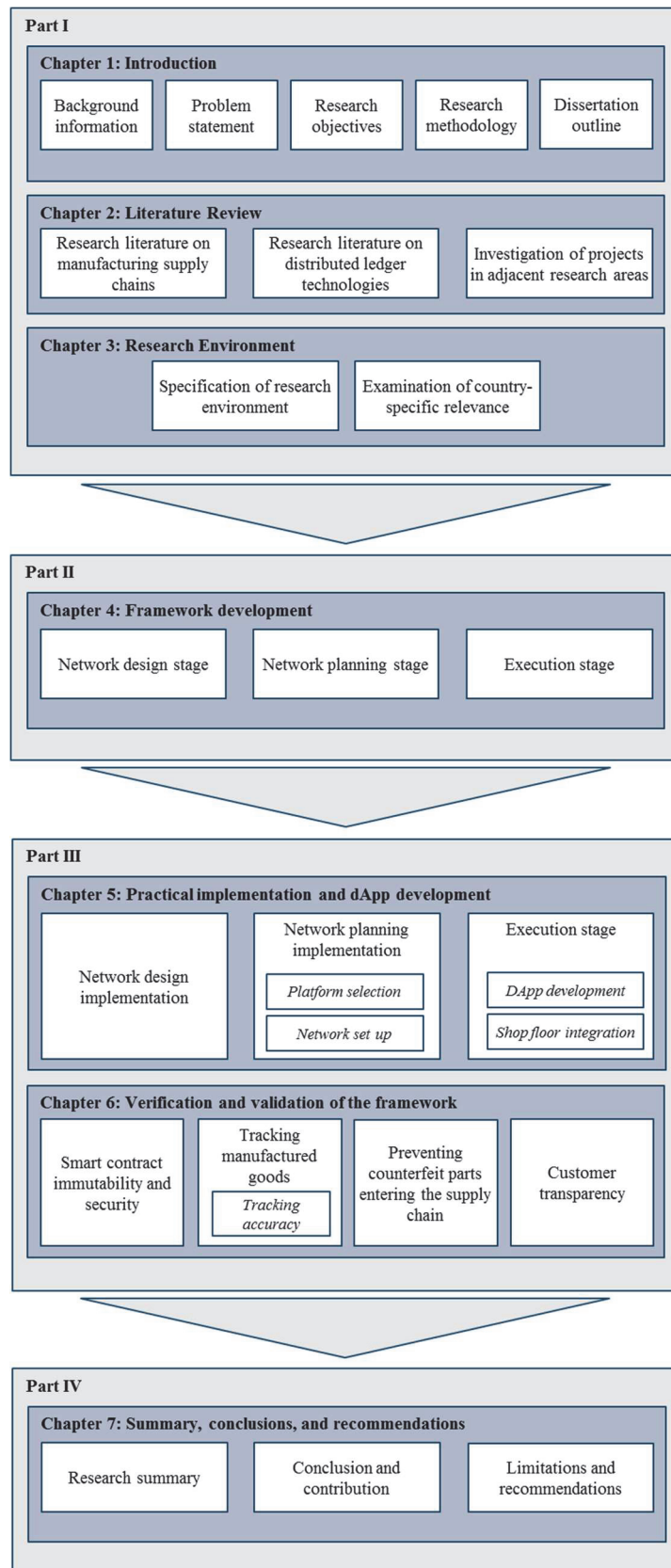


Figure 4: Structure of the thesis

2 Literature Review

The literature review is divided into three main parts. First, the chapter provides fundamental knowledge about manufacturing supply chains and distributed ledger technologies. At the end, the chapter describes existing projects that are already working on distributed ledger technologies in conjunction with supply chains.

2.1 Fundamentals: manufacturing supply chain

This section of the literature review focuses on the fundamental characteristics of manufacturing supply chains. First, the general structure of manufacturing supply chains and important definitions are introduced. Subsequently, the section reviews the main complexity drivers of manufacturing supply chains and the difference between supply chain transparency and visibility. In conclusion, the section describes the structure of supply chain management systems and their components.

2.1.1 General structure of manufacturing supply chains

Rapid changes due to globalisation and the increasing complexity of supply chains have a direct influence on the manufacturing landscape (Mourtzis & Doukas, 2012; Ueda et al., 2009). Over the last two decades product life cycles have become more than 30% shorter in some industries. Simultaneously, the variety of products has multiplied and ‘mass customisation’ has become much more scalable (Ernst & Haar, 2019).

According to Rudberg & Olhager (2003, p. 29), the vast majority of manufacturing is carried out in so-called ‘value’ networks, which are defined as networks of facilities, possibly owned by different organisations, where time, place or shape utility is added to goods in various stages such that the value for the ultimate customer is increased. Camarinha-Matos & Afsarmanesh (2006, p. 4) define a similar type of network as a ‘collaborative’ network. Therefore, collaborative networks consist of a variety of entities (e.g. organisations and people) that are largely autonomous, geographically distributed, and heterogeneous in terms of their operating environment, culture, social capital and goals, but that collaborate to better achieve common or compatible goals, and whose interactions are supported by computer networks. Based on these two definitions, this study considers manufacturing networks as collaborative value networks.

The typical structure of manufacturing supply chains usually consists of a network defined by a focal company, its suppliers, and the customers. In manufacturing supply chains such as the automotive industry, the focal company is also referred to as Original Equipment Manufacturer (OEM). The OEM and the suppliers provide all capabilities required to create and support the end products, which then can be purchased by the customers. When manufacturing operations become more and more specialised and complex, the OEM relies heavily on its suppliers, who are again relying on their own suppliers (National Research Council (U.S.), 2000, p. 23). Figure 5 shows a traditional manufacturing supply chain network structure.

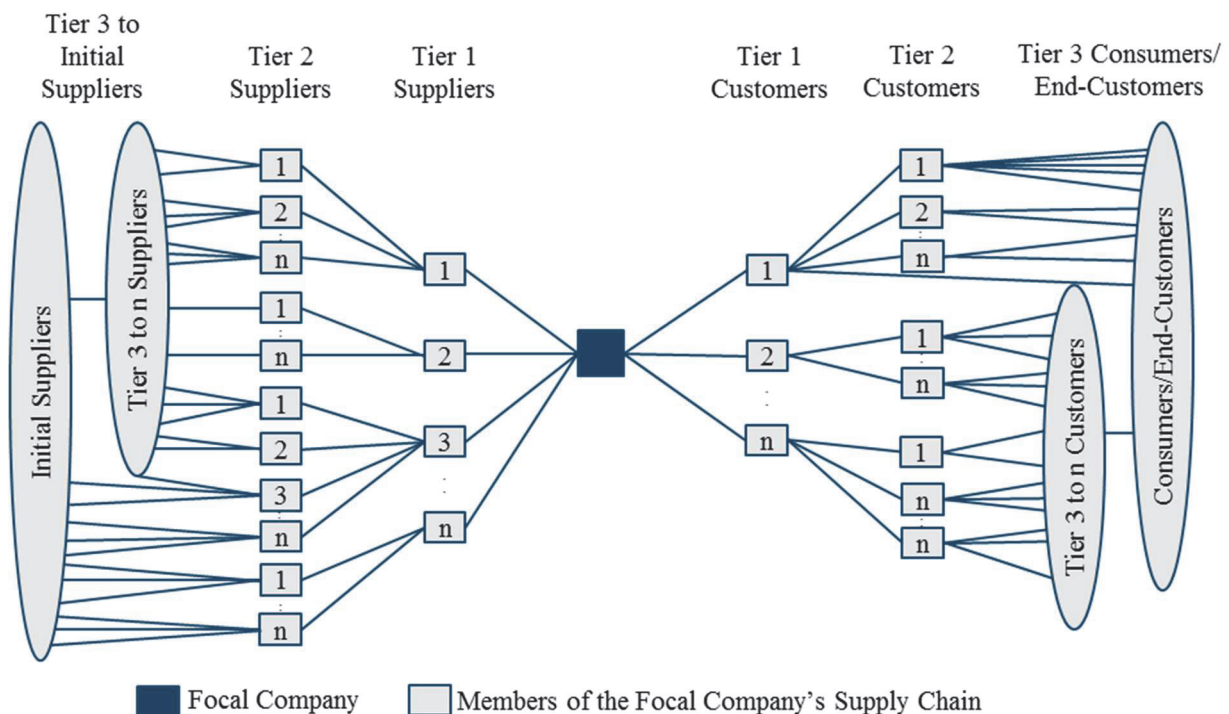


Figure 5: Network structure of a typical manufacturing supply chain (based on D. M. Lambert, Cooper, & Pagh, 1998, p. 3)

For a long time, global supply chains were characterised by the trade of finished products. Due to globalisation and the so-called ‘network trade’, global supply chains have become more and more fragmented. In addition, the degree of diversification in terms of the procurement of raw materials and components has increased. The distribution of procurement among different suppliers brings various advantages such as a wider choice, cost advantages and better risk diversification. At the same time, this strategy introduces completely new and significantly expanded requirements when planning, coordinating, managing, and controlling global supply

chains (Lehmacher, 2016, pp. 22–24). These changes have led to a shift from traditional non-modular manufacturing supply chains to an increased use of modular manufacturing supply chains. While in the non-modular configuration the final assembler does all the assembly work by itself, in the modular configuration the final assembler apportions product modules to intermediate sub-assemblers (S. J. Hu, Zhu, Wang, & Koren, 2008, p. 45). A comparison between the two configurations is shown in Figure 6.

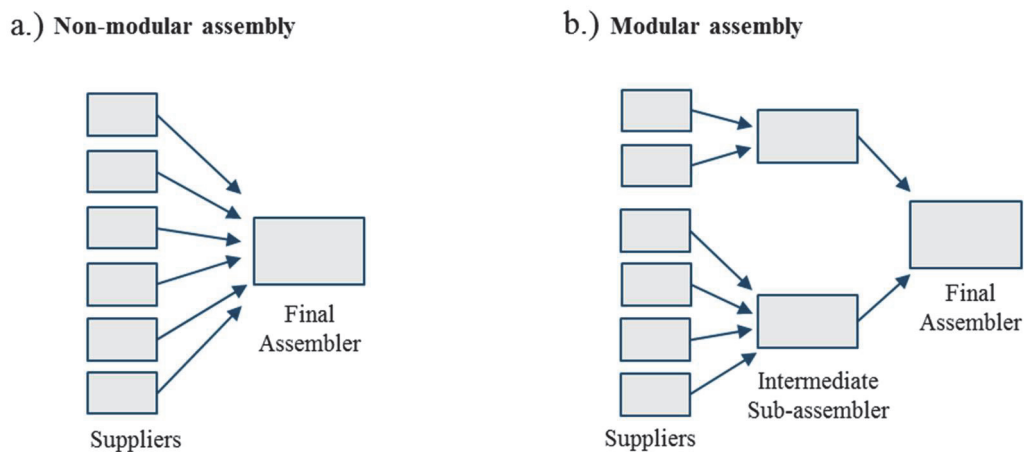


Figure 6: a.) Non-modular assembly; b.) Modular assembly (S. J. Hu et al., 2008, p. 46)

2.1.2 Complexity drivers of manufacturing supply chains

This section is mainly based on the results of a study conducted by Bozarth, Warsing, Flynn, and Flynn (2009), with the aim to determine the main complexity drivers of manufacturing supply chains in order to analyse their impact on the performance of manufacturing plants. For this work, the determination of variables impacting the supply chain's complexity is of special importance.

Bozarth et al. (2009) contrast two complexity types: detail complexity and dynamic complexity, whilst mixed forms of both types can also occur. Furthermore, they separate the complexity drivers into three main categories: Downstream complexity, internal manufacturing complexity, and upstream complexity. The definition of each type and category is listed below:

1. *Detail complexity*. The distinct number of components or parts that make up a system (Bozarth et al., 2009, p. 79).

2. *Dynamic complexity*. The unpredictability of a system's response to a given set of inputs, driven in part by the interconnectedness of the many parts that make up the system (Bozarth et al., 2009, p. 79).
3. *Downstream complexity*. The level of detail and dynamic complexity originating in the downstream markets of manufacturing supply chains (Bozarth et al., 2009, p. 81).
4. *Upstream complexity*. The level of detail and dynamic complexity originating the supply base of manufacturing supply chains (Bozarth et al., 2009, p. 81).
5. *Internal manufacturing complexity*. The level of detail and dynamic complexity found within the products, processes, and planning and control systems of the manufacturing supply chain (Bozarth et al., 2009, p. 80).

Table 3 summarises the results of Bozarth et al. (2009) according to the determination of key drivers of manufacturing supply chains' complexity.

Table 3: Main complexity drivers of manufacturing supply chains (based on Bozarth et al., 2009, p. 82)

Driver	Complexity type
<i>Downstream complexity</i>	
Number of customers	Detail
Heterogeneity in customer needs	Detail and dynamic
Shorter product life cycles	Detail and dynamic
Demand variability	Dynamic
<i>Internal manufacturing complexity</i>	
Number of products	Detail
Number of parts	Detail
One-of-a-kind/low volume batch production	Detail and dynamic
Manufacturing schedule instability	Dynamic
<i>Upstream complexity</i>	
Number of suppliers	Detail and dynamic
Long and/or unreliable supplier lead times	Detail and dynamic
Globalisation of the supply base	Dynamic

2.1.3 Supply chain transparency

Besides the structural complexity of global manufacturing supply chains, companies have to deal with the growing interest of all stakeholders in having a greater transparency of brands, manufacturers, and producers throughout the supply chain. Social and environmental sustainability issues in particular, have become increasingly important for manufacturers in order to maintain a flawless brand reputation. Meeting the transparency expectations of stakeholders is further complicated because supply chains spread across developing countries, where governments have only a limited ability and willingness to enforce their own laws. Therefore, the distributed nature of today's supply chains creates increasing levels of risk for multinational businesses, making transparency of supply chains both critical and complex (Chen, Zhang, & Zhou, 2018; Linich, 2014; New, 2010).

For Francis (2008, p. 182), supply chain visibility (SCV) is the identity, location and status of entities transiting the supply chain, captured in timely messages about events, along with the planned and actual dates/times for these events. Doorey (2011) extends the traceability aspect of supply chain visibility with a transparency aspect by the disclosure of all information of a product's flow throughout the production process and its supply chain to all stakeholders, especially the customers. Duckworth (2018) points out that in this context the terms visibility and transparency are frequently used interchangeably. In general, the term visibility focuses more on the data sharing within the supply chain to make a collaboration between the network partners more efficient. Transparency, however, often refers to the disclosure of information to all stakeholders, including the customer.

A lack of supply chain visibility not only puts the reputation of manufacturing companies at risk, but also poses a direct problem for the quality of the products itself. Due to the complexity of global supply chains, it becomes increasingly difficult to maintain the traceability of all components even from very reputable authorised distributors. This increases the probability of counterfeit components being introduced into a supply chain (Collier et al., 2019; DiMase et al., 2016; Pecht, 2013). In addition, transparency problems in the supply chain can have an impact on all business areas of manufacturing companies, particularly because such visibility problems create uncertainties, which in turn often result in stock buffers. These again can result in over-ordering, long lead times, and delivery delays (Christopher & Lee, 2004). Notably, for organisations operating in complex and dispersed supply chains such as manufacturing supply chains, the expansion of measures to increase the supply chain visibility can be of great advantage to reduce these risks (Brandon-Jones, Squire, Autry, & Pertersen, 2014; Linich, 2014).

2.1.4 Supply chain management systems

The management of global supply chains is more complex and dynamic nowadays than ever before. Therefore it requires properly configured and adaptable systems to facilitate the supply chain management (Ernst & Haar, 2019; Lehmacher, 2016, pp. 22–24).

For Kovacs & Paganelli (2003, p. 165), supply chain management (SCM) focuses on globalisation and information management tools, which integrate procurement, operations, and logistics from raw materials to customer satisfaction. For Serdarasan (2013, p. 533), the

increasingly complex nature of global supply chains adds to the difficulty of managing them to such an extent, that it almost becomes common sense to say that supply chain management is about managing the complexity of the supply chains. Christopher (2016, p. 3) defines supply chain management as the management of upstream and downstream relationships with suppliers and customers in order to deliver superior customer value at less cost to the supply chain as a whole. Christopher (2016, p. 3) also points out, that even supply chain management is a commonly used term, ‘demand chain management’ would be more appropriate, since it reflects the market-driven aspect of supply chains. Equally, the word ‘chain’ can be replaced by ‘network’ as there will normally be multiple suppliers as well as multiple customers to be included in the total system.

SCM software systems play a decisive role in the SCM. The task model of SCM software systems can be divided into three main levels: supply chain design, supply chain planning, and supply chain execution (Werner, 2017, pp. 86–93). Figure 7 visualises the simplified structure of the individual levels. In the following subsections, the levels and their components, which are of importance for this work, are described in more detail.

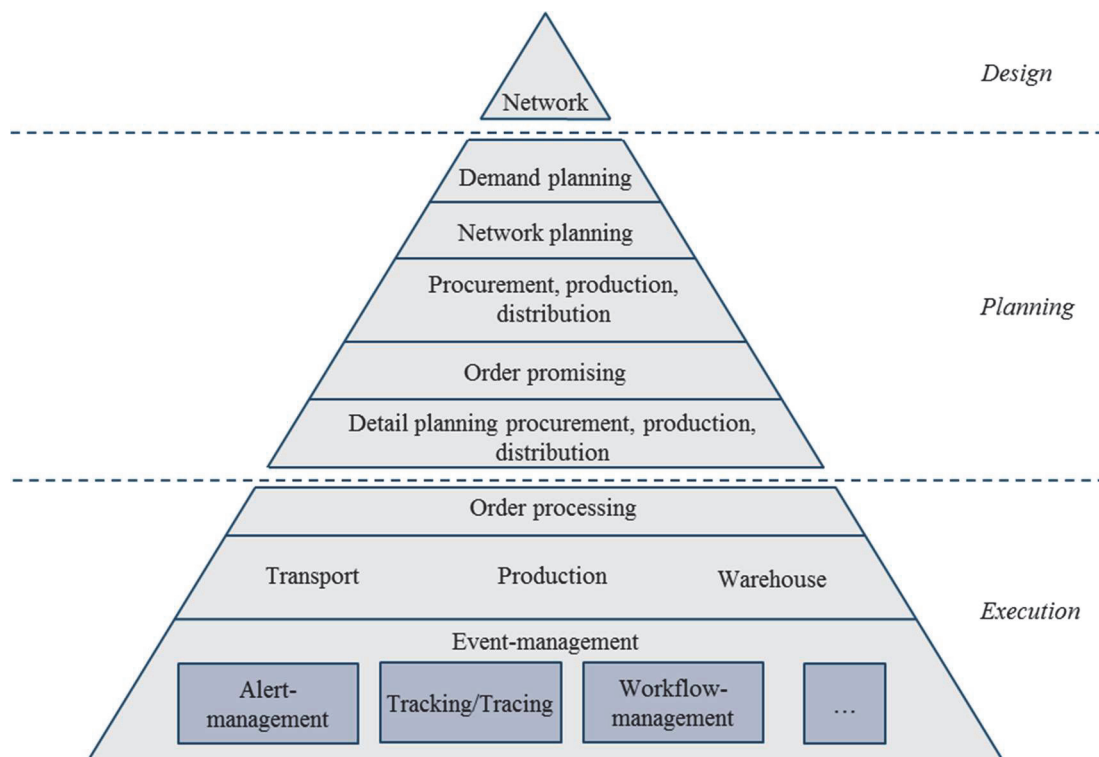


Figure 7: Task model of supply chain management software systems (based on Werner, 2017, p. 87)

2.1.4.1 *Supply chain design*

The supply chain design determines fundamental strategic investment decisions. On this basis, serious changes in terms of costs can occur within the entire supply chain. For example, the selection of a supply chain management software solution depends on the number of plants, suppliers, trading partners, distribution centres, or carriers involved (Werner, 2017, pp. 87–88). The decision-making during the supply chain design can be centralised or decentralised. In a centralised decision-making scenario, one decision-maker makes all decisions affecting the other entities in the supply chain. This scenario can be suitable for globally distributed networks. In a decentralised decision-making scenario, each entity can make its own decisions, which results in further challenges in both modelling and computation (Garcia & You, 2015, p. 163). It is also important during this phase, that a clear goal is already defined and recognised in the supply chain design. This presupposes that the kind of contribution each component of the system can make to the firm be analysed and that the investment then be planned as per the particular requirements of the organisation and the industry demand. Examples could be, for example, an increase in supply chain transparency, but also purely strategic approaches such as making the supply chain more sustainable (Eskandarpour, Dejax, Miemczyk, & Péton, 2015; Khan & Yu, 2019, pp. 255–256).

2.1.4.2 *Supply chain planning*

The second main level of the model is based on collaborative planning decisions within the supply chain which takes place after the strategic considerations of the supply chain design have been fixed. In the course of the supply chain planning, the planning of the tactical and operative implementation begins. In this respect, requirements, inventories, capacities or capacity allocations of all entities involved in the process must be aligned. One of the key components of the planning level is the network planning, where the coordination of individual actors in a supply chain takes place. Within the scope of internal planning, for example, the production and logistics centres of business areas must be defined worldwide. For distributed cross-company networks, procurement, production and distribution planning along the entire logistic chain in the collaborative network are clarified. Dominant entities occupy a special position in this respect because they have the most comprehensive information for planning, controlling and monitoring of the entire value chain (Werner, 2017, pp. 88–92).

In particular, in collaborative networks, it is important to define the scope of shared information among the supply chain members such as sales data, inventory, forecast and demand, production product planning, order status, transportation (Khan & Yu, 2019, p. 27). The information sharing among supply chain members can be divided into three main categories (Khan & Yu, 2019, pp. 27–29):

- 1.) *Product information*. This category includes the exchange and sharing of product-related information.
- 2.) *Transaction information*. This category includes the exchange and sharing of transaction information taking place in the network. This information can serve as a critical source in order to determine future trends but also to facilitate planning processes.
- 3.) *Information of inventory*. This category includes the exchange and sharing of inventory information including quantity, decision models, requirements, and all information affecting order processes. For many companies, this information is even more sensitive than transaction information. For example, full inventory transparency can have a strong impact on price negotiations among collaborative network partners.

2.1.4.3 *Supply chain execution*

After the planning activities have been completed, executing logistical activities is initiated. Important tools for executing logistical activities include, for example, alert management, workflow management and tracking and tracing systems (Werner, 2017, pp. 92–94):

- 1.) *Alert management*. Alert management is used to identify deviations between actual and target processes as early as possible. In particular, monitoring systems are ideal for alert management, which can react on predefined tolerance profiles.
- 2.) *Workflow management*. Workflow management is used to understand the electronic monitoring of work processes. The respective activities in workflow management are dependent on each other. A follow-up activity is controlled directly by the output of

the previous activity. If deviations occur, the information flow will automatically be interrupted.

- 3.) Tracking and tracing. Tracking and tracing describes the tracking and tracing of products and shipments. Customers are increasingly demanding to be integrated into the process of service provision. Therefore, one approach of the SCM is to integrate the customers into the tracking and tracing systems of companies (Werner, 2017, p. 160).

2.1.4.4 *Tracking and tracing systems to increase transparency*

Tracking and tracing describes the monitoring (tracking) and data archiving (tracing) of products or shipments (Werner, 2017, p. 513). Tracking and tracing systems depict an important component when increasing the supply chain transparency. Such systems are mostly applied for detection purposes. Devices attached to the goods send information about their status in pre-defined time intervals to allow various technologies to detect irregularities of processes (Eßig, Hülsmann, Kern, & Klein-Schmeink, 2013, p. 161). Two technologies play a significant role in order to provide tracking and tracing systems with information.

- 1.) *Barcode technology*. A barcode is a series of different width lines that can be presented in a vertical order (ladder orientation) or horizontal order (picket fence orientation). Barcodes developed from one to two dimensions and then to the latest QR codes (Quick Response codes). This standardised form makes barcodes appropriate for reading through computerised machines such as optical scanners. Barcodes represent an inexpensive and easy method to label products. Since barcodes can only be evaluated by reading devices, they are referred to as passive information carriers and thus do not have the ability to provide systems with real-time information (Khan & Yu, 2019, p. 250; Pundir, Jagannath, & Ganapathy, 2019, p. 158; Werner, 2017, p. 335).
- 2.) *RFID technology*. RFID (Radio Frequency Identification), is an automatic non-contact identification and communication technology. RFID tags can save and manage information about objects using a radio-frequency signal. The size of RFID tags depends on how many data strings can be stored and on the distance from which it can be read. Similar to barcodes, also RFID tags require RFID readers in order to evaluate the stored information. But unlike barcodes, the readers don't have to be in visual

contact with the tag. RFID tags can be divided into passive and active tags. Passive tags do not have their own power supply and thus have a lower reading range. Active RFID utilises tags with inbuilt batteries and they therefore have a higher reading range. This high reading range enables real-time information sharing as long as the power supply of the tag is guaranteed. Unlike barcodes, RFID represents a cost-intensive approach to solve the tracking problems of supply chains. (Feng, 2016, p. 2; Pundir et al., 2019, p. 158; Werner, 2017, p. 335).

Usually, the tracking of goods is mainly carried out by satellite-controlled tools such as GPS (Global Positioning System) and AEI (Automatic Equipment Identification). Barcodes and RFID are an important source of information in terms of informing the tracking systems about the goods' current status. This combination results in a holistic system with several advantages. For example, the administrative effort involved in data management decreases significantly. Furthermore, apart from warehousing, the fleet management of competitors is improving (Werner, 2017, p. 240).

The information from all components of supply chain tracking systems is usually sent to a central storage facility where they can be stored and evaluated. In doing so, companies are increasingly focusing on the use of 'cloud computing' (Vogel-Heuser, Bauernhansl, & Hompel, 2017; Werner, 2017, p. 240). The term cloud computing is used to describe information and data processing services provided over the Internet (Lehmacher, 2016, p. 186). Ideally, a global supply chain has its own cloud storage available to allow visibility and to ensure the authenticity and quality of what is captured. This allows for central storage of data, its evaluation, and finally access to distributed components, for example to import software updates or to make configuration changes. Cloud infrastructures include physically distributed systems to store and process data which are managed by a central cloud management. Therefore, the stored data and the services provided by the cloud can be stored distributed within the cloud infrastructure, but externally the cloud presents itself as a central service with a single entry point (Fallenbeck & Eckert, 2017, pp. 134–169; Lehmacher, 2016, p. 188). This structure of a centralised system results inter alia in less stability due to a central point of failure and hence in less security of the system (Singhal, Dhameja, & Panda, 2018, pp. 14–15).

2.2 Fundamentals: Distributed ledger technology

This section describes fundamental elements of distributed ledger technologies. Initially, the differences between centralised and decentralised are clarified. Following this, the blockchain technology, a special version of a distributed ledger technology, is described in more detail. Hashing, the transaction logic, consensus algorithms, and smart contracts represent essential components of this technology.

2.2.1 From centralised to decentralised systems

Decentralised systems represent the counterpart of conventional central systems. However, the terms decentralised and centralised are poorly defined and misleading in many places. That is mainly because there are hardly any systems that are purely centralised or decentralised. Additionally, centralised and decentralised systems can both be seen as distributed systems. Figure 8 shows an example of a physically distributed system with centralised control (such as clouds described in Section 2.1.4.4). Therefore, a centralised distributed system consists of a master node that is responsible for breaking down the tasks or data and distributing them across the nodes. A decentralised system on the other hand, gets along without a central master node and the system itself may distribute the required computation (Singhal et al., 2018, pp. 11–17). A node in general, is an individual player in a distributed system. All nodes are capable of sending and receiving messages to and from each other. Nodes have a memory and a processor and can be honest, faulty, or malicious (Bashir, 2018, p. 12).

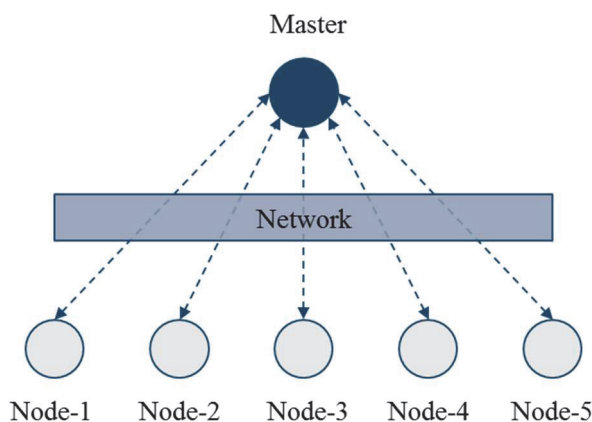


Figure 8: A physically distributed system with centralised control (based on Singhal et al., 2018, p. 12)

A centralised system (Figure 9) is characterised by the existence of a centralised control with all administrative authority. Such systems have benefits in terms of designing, maintaining, governing, and imposing of trust. At the same time, the structure of centralised systems results in the following inherent limitations (Singhal et al., 2018, pp. 14–15):

- a) Less stability of the system due to the existence of a central point of failure;
- b) Higher attack vulnerability and hence less security;
- c) Centralisation of power can lead to unethical operations;
- d) Usually the scalability is a difficulty.

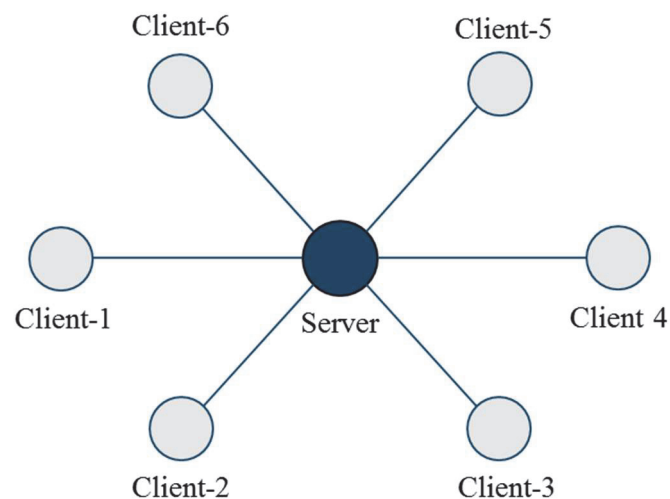


Figure 9: Typical structure of a centralised system (based on Berentsen & Schär, 2017, p. 96)

A decentralised system does not have a centralised control and every node has equal authority. This results in difficulties in terms of designing, maintaining, governing, and imposing of trust. However, the structure of decentralised systems offers the following advantages (Singhal et al., 2018, pp. 15–16):

- a) More stability and fault tolerance because there is no central point of failure;
- b) Higher attack resistance and thus more security;
- c) Symmetric system with equal authority to all, so less scope of unethical operations and usually democratic in nature.

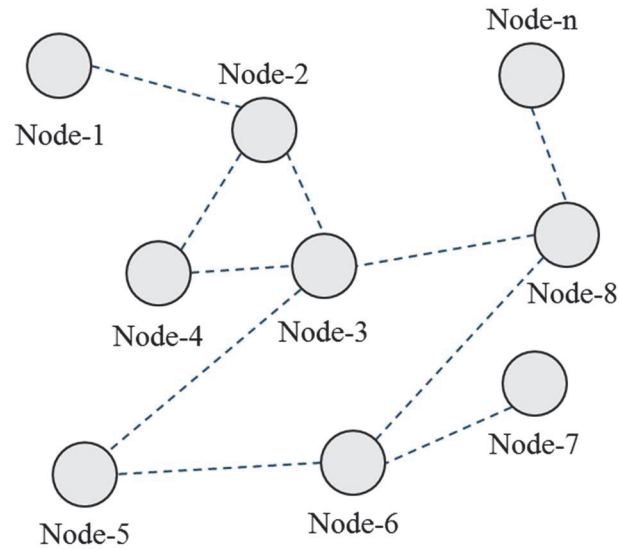


Figure 10: Structure of a decentralised system (based on Singhal et al., 2018, p. 16)

A special form of decentralised systems are peer-to-peer systems. Peer-to-peer (P2P) systems and applications are distributed systems without any centralised control or hierarchical organisations, where the software running at each node is not only equivalent in authority, but also in functionality. The contributing nodes work not only on a portion of the work, rather the interested nodes (or the randomly chosen ones) perform the entire work (Singhal et al., 2018, p. 16; Stoica et al., p. 1). Decentralised P2P systems as shown in Figure 11, are the foundation for distributed ledger technologies such as the blockchain technology (Nakamoto, 2008).

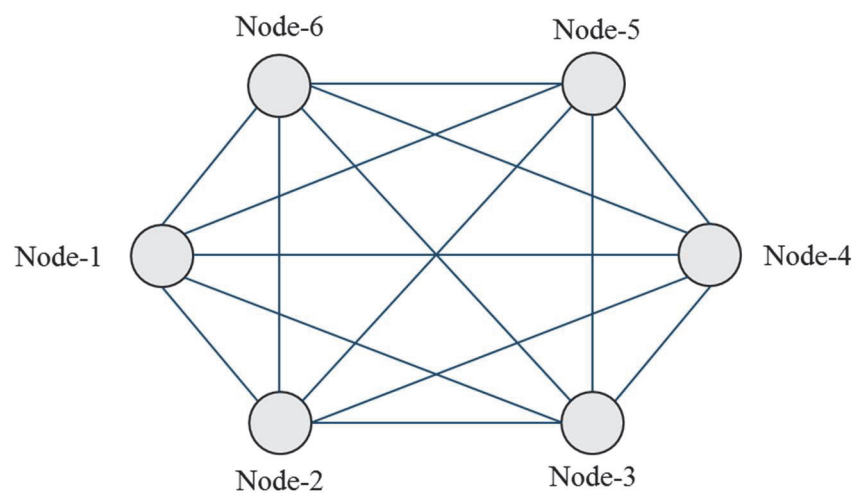


Figure 11: A decentralised peer-to-peer system (based on Singhal et al., 2018, p. 17)

Such peer-to-peer networks form the basis for so-called distributed ledger technologies (DLTs). In general a distributed ledger simply means that a ledger is spread across the network among all peers in the network, and each peer holds a copy of the ledger (Bashir, 2018, p. 17). When discussing DLTs, it is important to distinguish between different levels of distributed ledgers (Bott & Milkau, 2016, p. 157).

- a) *First level*: Technological synchronisation within distributed databases in a network.
- b) *Second level*: A unique (logical) ledger with bookkeeping of transactions and a replica at all participants.
- c) *Third level*: Contractual relations between the participants of the system.
- d) *Fourth level*: An overall perspective of ‘trust’ between participants (with the different points of view of anonymous participants, honourable merchants, legislation and enforcement, and of regulation).

Because of this wide spectrum of possible deployments of DLTs, this work will refer to the technology as combination of the fourth level and components including “peer-to-peer networking, distributed data storage, and cryptography that, among other things, can potentially change the way in which the storage, recordkeeping, and transfer of a digital asset is done” (Mills et al., 2016, p. 3).

The uniqueness of the DLT comes from two main characteristics. Firstly, the distributed nature and secondly because of the possibility for users to ‘deposit’ assets on the ledger digital (e.g. records, acts, and states). To ensure these properties, an agreement about the state of the ledger must be achieved by a consensus mechanism. This allows the network users to remove the need for a trusted third-party (socially constructed and sometimes flawed) intermediaries and replace it with ‘trust in the algorithm’. At the same time, using the distributed foundation of DLTs and the use of cryptographic algorithms, the record is rendered immutable, transparent, and auditable yet resistant to censorship and manipulation. In addition to the immutable recorded transactions, the system can provide a high level of anonymity or pseudonymity for transaction partners. Data records are visible at meta-level and remain tamper-resistant, but nevertheless, the identification of individual parties may be rendered difficult or impossible (Maull, Godsiff, Mulligan, Brown, & Kewell, 2017, pp. 483–484).

At its extreme, the architecture of DLTs is in contrast to conventional central systems (Section 2.2.1), allowing any entity (such as end users or financial institutions) to become a node in the network and share database management responsibilities directly with each other on a peer-to-peer basis. Figure 12 shows two examples of peer-to-peer connectivity. In both examples, each entity maintains a copy of a common ledger. Although the connectivity of the nodes is the same in both examples, the ownership and housing of the nodes is different. In example a.) all nodes are hosted within a single entity, while in example b.) multiple entities are in the arrangement and each of them hosts a node. The ability of an entity to participate in a DLT arrangement, depends not only on the computing resources of the entity, but on whether the arrangement is designed to be an open or closed system. Open systems accept all interested entities with the technical ability to participate. Closed systems, however, have additional membership criteria that must be met in order for an entity to be permitted to operate a node (Mills et al., 2016).

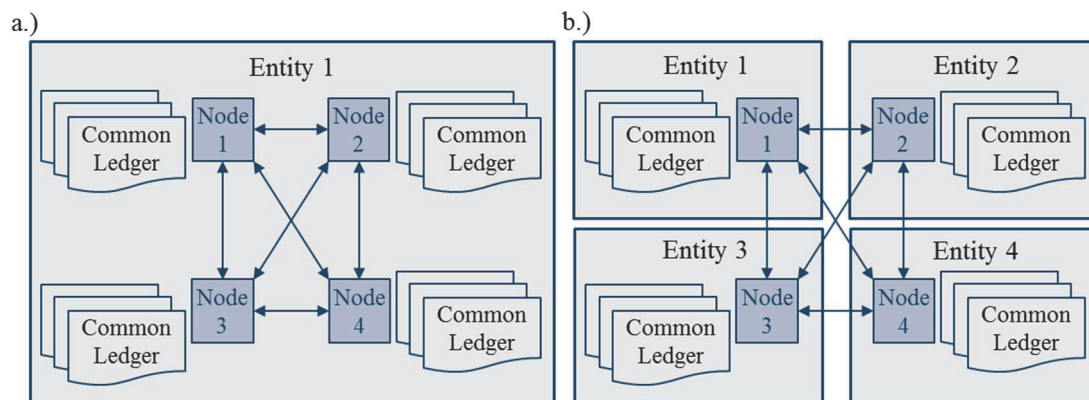


Figure 12: a.) Distributed ledger technology arrangement with nodes hosted within a single entity; b.) Distributed ledger technology arrangement with multiple entities, each hosting a node (Mills et al., 2016, p. 11)

DLT arrangements in which the participants are allowed to perform all activities are called ‘permissionless’ networks. Those that restrict participants’ activities are referred to as ‘permissioned’ networks. Combined with the design of open and closed systems shown in Figure 12, this generally results in three categories of DLTs: Permissionless public DLT; permissioned consortium DLT; and permissioned private DLT. In a public DLT, all records are visible to the public and everyone could take part at the consensus process. In a consortium DLT, only a group of preselected nodes would participate at the consensus process. As for a private DLT, only nodes coming from one specific organisation would be allowed to

participate in the consensus process (Buterin, 2015; Mills et al., 2016; Zheng, Xie, Dai, Chen, & Wang, 2017). The characteristics of each category is listed in Table 4.

Table 4: Comparisons among public, consortium, and private distributed ledgers (based on Zheng et al., 2017, p. 559)

Property	Public DLT	Consortium DLT	Private DLT
Consensus determination	All nodes	Selected set of nodes	One organisation
Read permission	Public	Could be public or restricted	Could be public or restricted
Immutability	Nearly impossible to tamper	Could be tampered	Could be tampered
Efficiency	Low	High	High
Centralised	No	Partial	Yes
Consensus process	Permissionless	Permissioned	Permissioned

Before the breakthrough of blockchain technology in 2009, several approaches to reach consensus in P2P networks existed, particularly to prevent the so called ‘double-spending’ problem in distributed electronic payment schemes (Hoepman, 2007; Osipktov, Vasserman, & Hopper, 2007). Hoepman (2007) even proposed several mathematical approaches to achieve a reasonable level of double-spending prevention to refute the commonly held belief that a prevention of double-spending is fundamentally impossible in P2P networks.

The ‘double-spending’ problem is the main security problem that DLTs have to deal with. Unlike physical currencies, where the physical money is hard to copy and once it has been spent it passes effectively to the recipients’ hands, digital currency tokens can be easily copied and double-spent if security mechanisms are not properly applied (Perez-Sola, Delgado-Segura, Navarro-Arribas, & Herrera-Joancomarti, 2018, p. 451). Since the release of the Bitcoin, a famous example of the blockchain technology, the prevention of the double-spending problem has been the foundation for many other token-based projects based on distributed ledger technologies such as IOTA, Byteball or Nano (Churyumov, 2016; LeMahieu, 2017; Nakamoto, 2008; Popov, 2018b).

2.2.2 Blockchain technology

The blockchain is defined as a technology to process and verify data transactions based on a distributed peer-to-peer network. The blockchain technology (BCT) is part of the distributed

ledger family. It uses cryptographic procedures, consensus algorithms, and back-linked blocks to make transactions practically unchangeable (Gentemann, 2019, p. 15).

In recent years, many blockchain-based projects or initiatives have experimented with new ways of raising funds through the issuance and sale of blockchain-based tokens. This procedure is also known as Initial Coin Offerings (ICOs) and still represents the most popular way to raise funds for blockchain-based projects (Hahn & Wons, 2018, p. 3; Primavera de Filippi et al., 2019). The approach of ICOs can be compared to an Initial Public Offering (IPO). The purely digital sale of tokens enables potential interested parties to participate at the project and its products ('utilities') at an early stage without establishing a direct (legal) relationship with the company itself (Hahn & Wons, 2018, p. 3). In literature, blockchain-based tokens can be divided into five different types (Hahn & Wons, 2018, pp. 8–15; Primavera de Filippi et al., 2019, pp. 9–12):

- 1.) *Digital Currencies*. A token of the type 'digital currency' or also known as cryptocurrency, is a purely digital store of value. Cryptocurrencies are integral to most public blockchains. Unlike government-issued 'fiat' currencies, cryptocurrencies are generally created in a decentralised fashion independently of any governments or central banks.
- 2.) *Utility Token*. As opposed to cryptocurrencies, utility tokens are issued on top of an existing blockchain infrastructure. A utility token has a certain functionality within a blockchain-based network or platform. The utility token is the most frequently used form of ICO. Most of these tokens are created on the Ethereum blockchain platform. Furthermore, as an integral component of the platform, the token itself has a direct functionality for the products and/or services offered there, and thus generates recognisable added value, for example in the form of software licences or access rights for decentralised applications.
- 3.) *Equity Token*. Unlike utility tokens, which are used or spent within an online platform, equity tokens are comparable to a share or an equity investment. Thus, the equity token is an investment in the company itself, often combined with profit-sharing and participation rights within the company.

4.) *Profit-share Token*. Unlike equity tokens, which confer the right to receive share capital or other forms of equity in the company, profit-share tokens confer the right to receive a payment only in the form of a dividend. Therefore profit-share tokens can be compared to a dividend payment on ordinary shares.

5.) *Asset-backed Token*. Asset-backed tokens represent physical assets and embody a ‘claim’ on the whole asset (for example gold, a painting, real estates) or a portion of such. Issuers could legally structure such a token in several ways.

In addition to the classification under a certain blockchain-based token type, hybrid tokens are also possible that meet two or more of the criteria listed above (Hahn & Wons, 2018, p. 12; Primavera de Filippi et al., 2019, p. 136).

As described at the end of Section 2.2.1, the double-spending problem is the main security problem DLTs have to deal with. In case of the BCT, this problem refers to the double-spending of tokens. To accomplish a solution without a trusted third party, the BCT uses a system for participants to agree on a single history of the order in which the transactions were received. Therefore it needs proof that the time of each transaction with the majority of nodes agreeing it was the first received (Nakamoto, 2008). As the name blockchain emphasises, the transaction history is arranged in blocks. Each block consists of a collection of data and references the block preceding it (Elrom, 2019, p. 9). This connection between the blocks results in a chain-like arrangement shown in Figure 13.

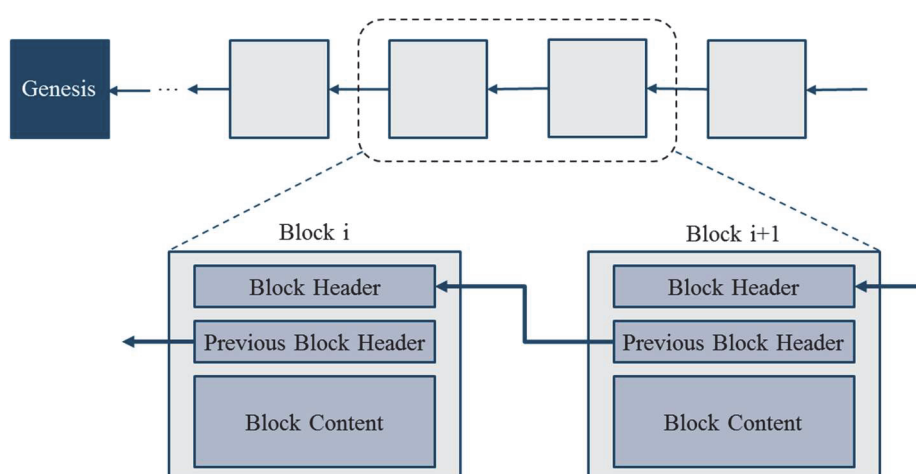


Figure 13: Linked blocks by reference to the previous block header hash (based on Dinh et al., 2018, p. 1367)

Figure 13 exemplifies that the first block generated on the blockchain, the genesis block, has an impact on all following blocks. Nevertheless, it only represents a simplified illustration of blocks in BCTs. In more detail, blocks of BCTs consist of a header of the current block, the header of the previous block, the content of the block summarised in the form of merkle-trees, and a so called ‘nonce’ value (Dinh et al., 2018, p. 1367). In order to understand the structure of BCT entirely and to create an important foundation for the following chapters, the basic function of BCTs is clarified in the following subsections. At the end, a summary presents a holistic picture of the structure of common BCTs.

2.2.2.1 Hashing in BCTs

In general, a hash is the outcome of an input by conducting a computation with regards to a mathematical hash function. Typically, hash functions use deterministic algorithms, meaning that the same input always creates the same output. Hash functions map a bit of string of arbitrary length to a fixed length string. It is statistically impossible to reverse this process in order to determine the input (Brennan & Lunn, 2016, pp. 19–20; Dang, 2012, p. 4). Famous BCTs such as Bitcoin usually refer to the Secure Hash Algorithm with a 256-bit length (SHA-256) consisting of 64 hexadecimal characters, designed by the National Security Agency (NSA) (Dang, 2012, pp. 6–7). The SHA-256 is a so-called cryptographic hash function, with different types of functions depending on the used mathematics. These algorithms make it not only harder to recreate data from a hash, but also even minor changes of the input alter the output in a much more significant way (Anton Badev & Matthew Chen, 2014, p. 9). Table 5 shows how minor changes of the input result in totally different SHA-256 hashes.

Table 5: Minor changes of the input result in totally different hashes

Input	SHA-256 Hash
Hello World	A591A6D40BF420404A011733CFB7B190D62C65BF0BCDA32B57B277D9AD9F146E
Helo World	5486BEC2F052F0E6B4E2797E611B0A3CBD3FE9F33E0B2494D4997B54653E1FD9

In order to outline the cryptographic capabilities of the SHA-256 and to illustrate the statistical impossibility to reverse the hashing process to determine the input, Brennan & Lunn (2016) simulate a brute force attack of the Bitcoin network. For this test it is assumed that the system is able to operate with the processing power of the entire Bitcoin network. The brute force approach simulates a fraud attempt by exchanging input data without any effect on the value

of the block header. As a result, the calculation would take almost $1.33 \cdot 10^{51}$ years to find an appropriate hash value. In other words, it would take almost 9,672,989,162 trillion times the age of the universe to successfully manipulate one single block.

The Ethereum platform uses a combination of two different hashing algorithms, which are based on very similar operating principles to the SHA-256. On the one hand, Ethereum uses the own Ethash algorithm and on the other the Keccak-256 algorithm. Ethereum hashes can be easily distinguished from hashes of other platforms because they always begin with '0x' to quickly identify them as hexadecimal numbers (Wood, 2019).

2.2.2.2 *Transaction logic*

Besides the immutability of data, the BCT offers further dynamic components to guarantee a high level of integrity for transactions. As described in Section 2.2.2, BCTs are particularly suitable for carrying out transactions of different tokens. In the case of cryptocurrency tokens, such transfers can be compared with online money transfers. So the transfer includes all relevant data to initiate the shared ledger to update the accounts of the users involved. On one side the balance is reduced to the number of tokens sent, on the other it is raised to the number of tokens received (Brennan & Lunn, 2016, pp. 20–23). Depending on the platform, transactions can also be executed by messages between users or simply by users interacting with the network. In a figurative sense, any change in the state of the distributed ledger results in a transaction that can either be accepted or denied by the network (Wood, 2019, pp. 4–5). In order to ensure the accuracy of these interactions, digital signatures between unknown participants are used. Therefore, BCT require the embedding of three necessary algorithms:

- 1.) *Key generation algorithm*. This algorithm creates two keys that inhere in a mathematical relationship; a private one for encryption and a public one for decryption of hashes. Based on advanced cryptography, they cannot be derived from each other (Brennan & Lunn, 2016, p. 20).
 - a.) The *public key* is a publicly known alphanumeric string that is hashed with another, privately held, string to sign a digital transaction. In case of BCT, the public key can also be seen as the address of an account (Olleros & Zhegu, 2016, p. 253).

- b.) The *private key* is an alphanumeric string kept secret by the user and is designed to sign a digital transaction when hashed with a public key. In case of BCT, only the owner of an account's private key is able to initiate transfers with this account (Olleros & Zhegu, 2016, p. 253).
- 2.) *Signing algorithm*: This algorithm initiates a computation that hashes the original message. By using this hash as an input, a digital signature is created by the sending party applying a private key for encryption (Brennan & Lunn, 2016, p. 20).
- 3.) *Verification algorithm*: By decrypting the signature through a public key and hashing the original message at the receiving side once more, a comparison between the two hashes takes place to verify if the transfer has been successful or not (Brennan & Lunn, 2016, p. 20).

In order to be able to execute transactions on a blockchain network, a user must have access to a 'wallet'. A blockchain wallet is a piece of software that enables users to have access to the blockchain platform. By connecting the wallet with the public and private key, users have the opportunity to execute transactions on the platform (Bruehl, 2017, p. 136).

Figure 14 shows a possible embedding of these algorithms to execute transactions on blockchain networks, as first described by Nakamoto (2008). As a first step, the key generation algorithm creates a pair of keys (public key and private key) for the sender. To execute a transaction, the sender needs the public key (address) of the receiver. As a second step, the sender generates a transaction according to the platform's transaction format. The signing algorithm then creates a signature of the data with the private key of the sender. These encrypted data can be sent to the public key of the receiver. Finally, the verification algorithm allows the receiver to verify the transaction, because the signature can only be generated by the owner of the private key that matches the public key of the sender.

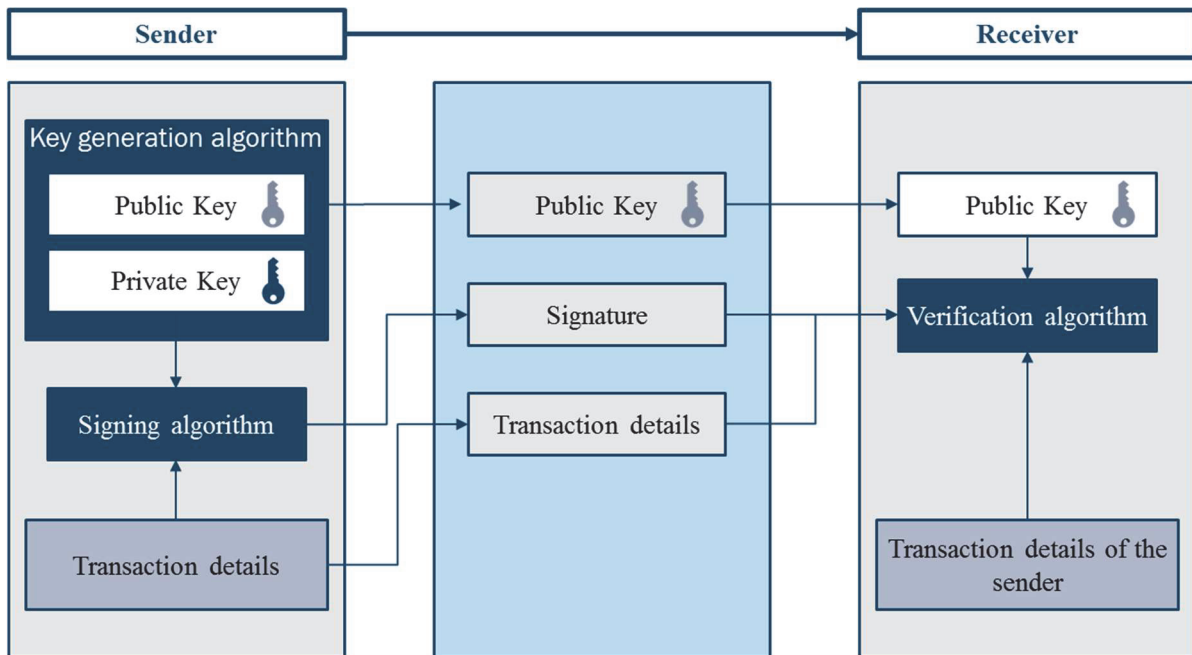


Figure 14: Workflow of blockchain transactions (based on Bruehl, 2017, p. 136)

To determine the accurate time of each transaction, each transaction is marked with a timestamp on the blockchain network (Zheng et al., 2017, p. 558). A timestamp in general, is a sequence of characters or encoded information identifying exactly when a certain event occurred. Trusted timestamping is the process of securely keeping track of the creation and modification of an event, which means that no one should be able to change it once it has been recorded. Especially concerning the decentralised BCT, it is important that the integrity of timestamps is never compromised (Lee & Deng, 2018, p. 168).

Transactions are grouped within the blocks of a blockchain into so-called 'merkle-root-hashes'. In this way, each transaction on the blockchain is hashed according to the hashing procedure described in Section 2.2.2.1. For this purpose, the individual hashes of the transactions are bundled in pairs until no further pair can be bundled, the so-called merkle-tree is created (Figure 15). The last bundle represents the merkle-root-hash. This linkage of transactions leads to incontrovertible evidence of every transaction that has occurred on the network (Rosenberger, 2018, pp. 67–68).

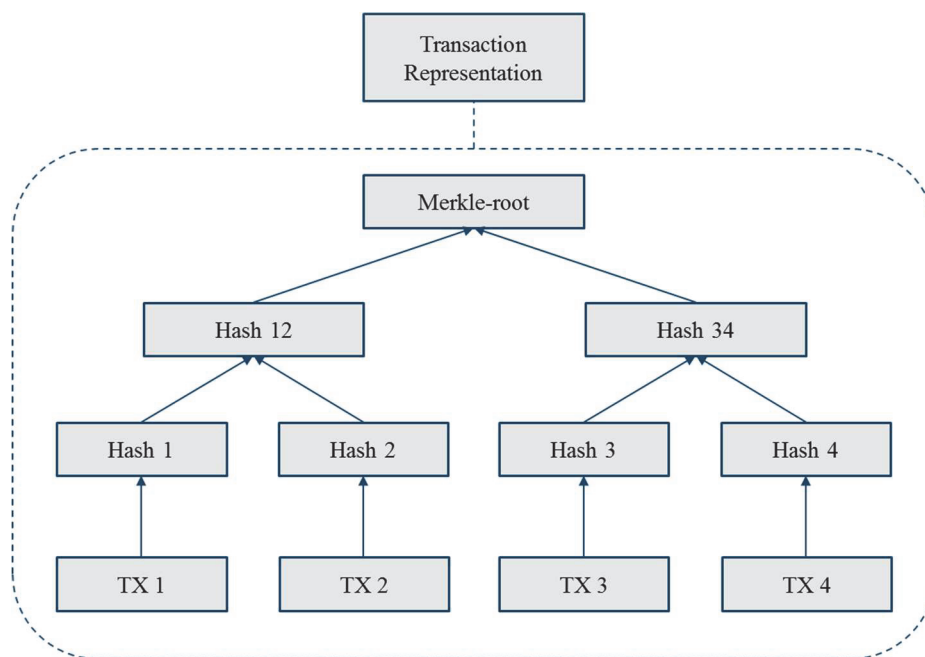


Figure 15: Transactions (TX) hashed in a merkle-tree (based on Wang et al., 2019, p. 22332)

The problem of the transaction verification process is that receivers of blockchain transactions are only able to verify if transactions come from valid senders or not. However, receivers cannot verify if the received transaction is actually double-spent. Due to the lack of a trusted third party in BCTs, it requires an alternative system to reach consensus about a single history of all transactions on the network in order to avoid the double-spending problem (Nakamoto, 2008).

2.2.2.3 Consensus algorithm

Consensus algorithms are algorithms used to maintain data consistency in distributed networks. They represent an essential function of the BCT in furtherance of achieving decentralised networks and to replace trusted third parties (Zheng et al., 2017, p. 558). A blockchain updating protocol is said to achieve consensus if the following properties are satisfied (Bano et al., 2017):

- 1.) *Validity*. All nodes activated on a common blockchain state propose to expand the blockchain by the same block. Additionally, and node transiting to a new local replica state of the blockchain, adopts the blockchain headed by that block.
- 2.) *Agreement*. When a node confirms a new block, then any other node updating its local blockchain will update it with the new block.

- 3.) *Liveness*. All transactions executed by nodes will eventually be confirmed.
- 4.) *Total order*. All nodes accept the same order of transactions as long as they are confirmed in their local blockchain.

The consensus algorithms vary with different blockchain platforms. In the following subsections, the most relevant approaches to reach consensus on blockchain networks are presented.

2.2.2.3.1 *Proof-of-Work (PoW)*

To add blocks to the blockchain, a node has to prove that it has performed some computational work, also known as Proof-of-Work (PoW). The core idea of this algorithm is to allocate the accounting rights and rewards through a hashing power competition among the nodes. Based on the information of the previous block, the different nodes calculate the specific solution of a mathematical problem. This mathematical problem is often referred to as mathematical puzzle. The first node solving this mathematical problem can create the next block (Baliga, 2017, p. 6; Mingxiao, Xiaofeng, Zhe, Xiangwei, & Qijun, 2017, p. 2568; Nguyen & Kim, 2018, p. 106). The PoW consensus algorithm is inter alia used by the blockchain platforms with the highest market capitalisation, Bitcoin and Ethereum (Buterin, 2014a; Nakamoto, 2008), and is therefore occasionally referred to as the original blockchain consensus algorithm (Nguyen & Kim, 2018, p. 106).

An important component of the PoW is the so-called ‘difficulty’. The difficulty level is dynamically tuned by the blockchain platform’s protocol depending on the computational power of the network to ensure that average production time of a block (block time) remains the same. The difficulty regulates how difficult it is for the nodes, to find a solution for the mathematical problem (Baliga, 2017, p. 6; Mingxiao et al., 2017, p. 2568). The simplified calculation steps are as follows:

- 1.) *Get the difficulty*. First, all nodes participating at the mathematical problem solving process, also known as ‘miners’ (Kiayias, Koutsoupias, Kyropoulou, & Tselekounis, 2016; Liu, Wang, Niyato, Zhao, & Wang, 2018), request the current difficulty of the platform’s protocol. As described in Section 2.2.2.1, hashes usually consist of 64 characters representing a hexadecimal number. In fact, the difficulty determines how

much smaller the value of the target hash must be compared to the hash value of the previous block (Mingxiao et al., 2017, p. 2568).

- 2.) *Collect transactions.* As the second step, the mining nodes collect all pending transactions on the network. Those transactions are summarised in a merkle-tree to define the merkle-root according to Figure 15 (Section 2.2.2.2). This leads to two fixed hash values in a block: the previous block header and the root merkle-root. Moreover, a variable ‘nonce’ value is added to these hashes (Mingxiao et al., 2017, p. 2568).
- 3.) *Calculating.* In this step, the computation work takes place. The miners change the variable nonce value until the hash value of the new block hash is less than or equal to the target value. If this is the case, the miner who found a ‘winning’ block, can broadcast it to the network (Baliga, 2017, p. 6; Mingxiao et al., 2017, p. 2568).
- 4.) *Restarting.* If a node can't work out the target hash value at a certain time, it repeats the process from step two. If any other node completes the calculation, then all nodes restart from step 1 (Mingxiao et al., 2017, p. 2568).

Due to the distributed, concurrent nature of this process, there is the possibility of two nodes finding a winning hash at the same time. In this case, each winning node adds its own proposed block to the blockchain and broadcasts this over the peer-to-peer network. Therefore, this results in a temporary fork in the blockchain, where some nodes can add blocks to the one branch, while other nodes are adding blocks to other branches. However, as more blocks are added to these forks, the platform’s protocol ensures that at some point the longest branch will get included in the main chain and others will be discarded (Baliga, 2017, p. 6).

The structure of the PoW makes this algorithm vulnerable to the so called ‘51% attacks’, where a miner controls more than 50% of the network’s computation power. In this way, the attacker can ensure to always be the first one creating the latest block and therefore can selectively include or reject the transactions which are processed into a block. However, due to the huge amount of computation power required to successfully execute such an attack, the PoW can be seen as an effective guarantee for the safety of blockchain networks (Baliga, 2017, p. 6; Mingxiao et al., 2017, p. 2568).

2.2.2.3.2 Proof-of-Stake (PoS)

The PoW consensus algorithm helped the BCT to reach a major breakthrough, however the nature of PoW also results in significant energy consumption (Baliga, 2017, p. 6). This problem resulted in the idea of an alternative algorithm called Proof-of-Stake (PoS), firstly introduced by Sunny King and Scott Nadal (2012). According to their approach, the main intention of the PoS is to reduce the energy consumption significantly while retaining the decentralised characteristics of the PoW. The PoS is used by various blockchain platforms such as EOS, Cordaono, Nebilo and PIVX (EOS Whitepaper, 2018; Jakiman, 2017; Kiayias, Russell, David, & Oliynykov, 2017; Nebilo Whitepaper, 2017). Due to the drastic reduction in energy consumption, Ethereum is also considering switching their network from the currently used PoW to a PoS algorithm in the future (Buterin, 2014b).

In literature several variants of PoS algorithms exist. The basic idea behind each PoS approach remains the same: PoS replaces the mining operation by involving a user's stake or ownership of virtual currency on the blockchain platform (Nguyen & Kim, 2018, pp. 110–111). Depending on the stake of a user, the PoS algorithm pseudo-randomly selects validators for the creation of a new block. Thus it is important to ensure that no validator can predict its turn in advance (Baliga, 2017, p. 8). Every miner stores a certain amount of tokens (stake) on the blockchain. The more stake a user owns, the higher probability they have to become the validator for the next block. Once a miner has successfully confirmed a new block, the validator receives the transaction fees as a reward. Conversely, a validator trying to defraud the network, will be penalised by losing part of their stake (Bentov, Gabizon, & Mizrahi, 2016).

Even though this PoS algorithm leads to great advantages in terms of energy consumption, they still suffer from other vulnerabilities. Since the amount of the stake is decisive for the probability of being selected as a block validator, 'rich' miners have better chances to create new blocks and receive the transaction fees. These transaction fees can then be reinvested into their stake, which in return further increases the chance of being selected to validate the next block. In addition to the moral aspect of PoS promoting 'rich' miners, this aspect can also cause technical problems known as 'nothing-at-stake'. Thus, it may happen that in a PoS blockchain two valid blocks are created at exactly the same time, which results in a temporary fork of the blockchain. The PoS algorithm allows miners to vote on multiple

forked versions of the blockchain, without splitting up the resources required to take part at the vote. For the miners, such a course of action is not combined with any financial risk. Therefore the miners have ‘nothing-at-stake’ while voting on multiple versions of the blockchain in order to maximise their chances of winning the transaction fees (Baliga, 2017, p. 8).

Similarly to the PoW, the PoS can also be affected by the 51% attack. In order to successfully complete an attack on the blockchain, an attacker has to control more than 50% of the network’s resources. In PoW approaches, the key resource allowing users to take over the system is the total computational power. In PoS approaches however, the key resource is the currency itself. Whoever owns over 50% of the total tokens available on the network, is able to manipulate the blockchain (Watanabe et al., 2016, p. 467).

2.2.2.3.3 Practical Byzantine fault tolerance

The Practical Byzantine Fault Tolerance (PBFT) mechanism is an effective approach to reach consensus in distributed systems, proposed firstly by Castro and Liskov (1999). Later, this mechanism was adapted to the BCT and today finds its use for example in the permissioned platform Hyperledger (Sukhwani, Martinez, Chang, Trivedi, & Rindos, 2017). A modified version of the PBFT, the so-called Delegated Byzantine Fault Tolerance (DBFT) algorithm, is used inter alia on the permissionless platform named NEO (NEO Whitepaper, 2019). The original PBFT consists of five stages (Castro & Liskov, 1999; Mingxiao et al., 2017, pp. 2568–2569):

- 1.) *Request*. The client sends a request to a master server node which marks the request with a timestamp.
- 2.) *Pre-prepare*. The master server node records the request messages and puts them into an order. Then the master node broadcasts a pre-prepare message to the other following server nodes, which initially determine whether to accept the request or not.
- 3.) *Prepare*. If a server node accepts a request, it broadcasts a prepare message to all the other server nodes and simultaneously receives the prepare messages from

other nodes. After collecting a determined number of messages the server node enters the commit state, if a most of the nodes accepted the request.

- 4.) *Commit*. Each node being in the commit state sends a commit message to all the other nodes in the server. At the same time, if a server node receives a determined number of commit messages, it is likely to believe that most nodes reach a consensus to accept the request. Then the node executes the instructions in the request message.
- 5.) *Reply*. The server nodes reply to the client. If the client does not receive a reply due to a network delay, the request is resent to the server nodes. If the request has been executed, the server nodes only need to send the reply message repeatedly.

When applying the original PBFT to the BCT, PBFT algorithms differentiate between leader nodes and validating nodes. The procedure, however, remains very similar. First, the clients send their requests for transactions to validating nodes. Then the receiving nodes validate the transactions and broadcast them to other nodes, including the leader node. After a determined number of transactions, or a certain time interval, is reached, the leader node puts the transactions into an order according to their created time and puts them into a block. Afterwards, three stages pre-prepare, commit, and prepare are executed. In the Pre-prepare phase, the leader broadcasts his proposed block to other nodes. These nodes will receive and store the proposed block locally. In order to make sure that the received block from the leader is the same, they execute a double-check by broadcasting the proposed block in the prepare stage and commit stage. If more than $2/3$ of all the nodes receive the same block, which they have already stored locally, they will execute the commit stage. After the commit stage is reached, the same procedure is repeated, which at the same time is the requirement for any node to execute the transactions in the proposed block in order to append them to their current blockchains (Nguyen & Kim, 2018, p. 117; Sukhwani et al., 2017).

PBFT assumes that node identities are known, therefore it can only work reliably in the permissioned networks. Furthermore, distinction between two different types of node does not reflect the basic idea of decentralised permissionless networks, in which all nodes have the same rights. Additionally, PBFT is unlikely to be able to scale to the network size of Ethereum, because of its high communication expense (Dinh et al., 2017, p. 1088).

2.2.2.4 *Smart contracts*

In 1994, Nick Szabo introduced the concept of smart contracts and described a smart contract as “a computerized transaction protocol that executes the terms of a contract. The general objectives of smart contract design are to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement), minimize exceptions both malicious and accidental, and minimize the need for trusted intermediaries” (Szabo, 1994). Szabo (1997) suggested to utilise protocols and user interfaces to translate contractual clauses into code and to embed them into hardware and software to facilitate all steps of the contracting process.

In 2013, Vitalik Buterin published the Ethereum white paper titled “A next generation smart contract & decentralized application platform” and therefore embedded Turing Completeness into the BCT (Buterin, 2013). Turing Completeness is a mathematical concept and is a measure of the computability of a programming language. Therefore, the language design includes complex constructs such as loops and conditions which enable it to create all types of general purpose programs (Lee & Deng, 2018, p. 155). Thus, Buterin (2013) coined the term smart contract with blockchain-based applications.

The code of each smart contract is stored on the blockchain and can be identified by a unique address. Users can interact with a smart contract in present cryptocurrencies by sending transactions to the contract address. When a user causes a valid new transaction with a smart contract address as recipient, all participants on the mining network execute the contract’s code with the current state of the blockchain and the transaction’s content as inputs. The network then agrees on the output and the next state of the contract by participating in a consensus protocol (Luu, Chu, Olickel, Saxena, & Hobor, 2016, p. 254).

Public blockchains ease all users to deploy ‘public smart contracts’, which has attracted a wide variety of commercial applications. Smart contracts on a permissioned blockchain or ‘permissioned smart contracts’, are more often used in collaborative business processes since they have the potential to prevent unwanted updates, improve efficiency and save costs. (Y. Hu et al., 2018, p. 3). The characteristics of public and permissioned smart contracts are shown in Table 6.

Table 6: Characteristics of public and permissioned smart contracts (Y. Hu et al., 2018, p. 3)

Type of contract/ Characteristic	Permissionless smart contracts	Permissioned smart contracts
Common	<ul style="list-style-type: none"> • Immutable record • Proper encryption on data and pseudonymity • Interoperability among different platforms • Traceable modifications 	
Unique	<ul style="list-style-type: none"> • Easy to deploy • Accessible for the public 	<ul style="list-style-type: none"> • Faster settlement • Lower operational cost • Permissioned access

It is possible to expand smart contracts to decentralised applications (dApps). A dApp is an application with graphical user interface (GUI), which uses smart contracts on the back end, in lieu of a conventional database and web application-hosting provider (Dannen, 2017, p. 77). DApps differ from smart contracts in two ways. First, a dApp has an unbounded number of participants on all sides of the market. Second, a dApp is not necessarily a financial application (Buterin, 2014a). In general, one can differentiate between three different types of smart contract applications (Buterin, 2013):

- 1.) *Financial applications*. Financial applications provide users with powerful ways of managing and entering into contracts using their money. Examples for financial applications are sub-currencies, financial derivatives, hedging contracts, savings wallets, wills, and ultimately even some classes of full-scale employment contracts.
- 2.) *Semi-financial applications*. Semi-financial applications are applications where money is involved but the application also comprises a significant non-monetary side. An example for semi-financial applications are contracts that reward users financially for solving mathematical problems.
- 3.) *Non-financial applications*. Non-financial application do not include any financial aspects. Typical examples for non-financial applications are online votes and decentral governances.

A typical dApp consists of three elements: Contracts and logic on the blockchain, user interface, and backend resources such as off-blockchain storage. The user interface is the only component loaded on a user device. The interface makes back-end calls to the blockchain to execute a particular contract and to the back-end resources if external storage is needed or in case the application needs to communicate with other apps. Figure 16 shows the typical structure of a dApp.

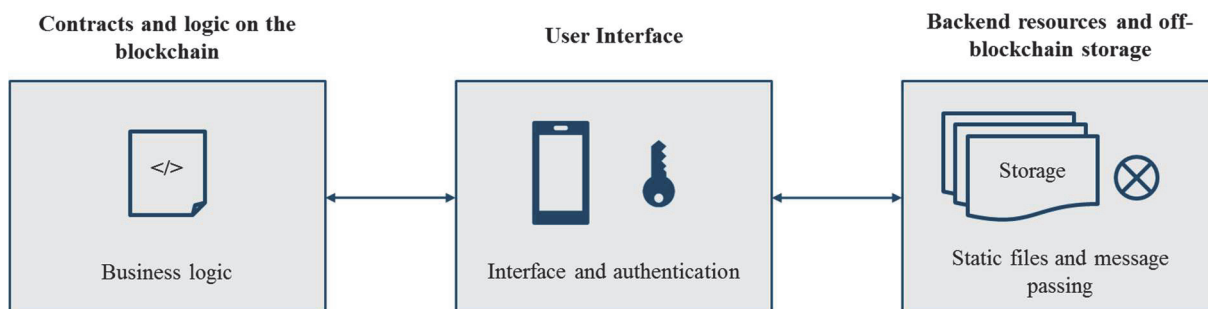


Figure 16: Structure of a decentralised application (Dhillon, Metcalf, & Hooper, 2017, p. 42)

Especially on public blockchain networks, dApps unfold their full effects and differ the most from traditional central applications. Therefore, dApps pertaining on a public blockchain should meet the following three criteria (Chohan, 2019):

- 1.) *Open Source*. The code of dApps must be completely open-source in order to reside on the decentralised blockchain architecture
- 2.) *Autonomy*. It is necessary that dApps are operating autonomously and no entity can control the majority of their tokens and thus monopolise their value or function. However, a dApp can still adjust its protocol if a consensus-based decision is made.
- 3.) *Cryptography*: A dApp must store its data and records cryptographically on a decentralised public blockchain. The token of the dApp must also be cryptographic, but not necessarily a native token. The cryptographic token ensures access to the dApp and serves as a reward to token miners, while also acting as a proof of value.

Depending on the platform, smart contracts can be programmed in several programming languages. Gavin Wood (2014) invented the object-oriented contract language Solidity, a

language that is specifically designed to target the smart contract development on the Ethereum Virtual Machine (EVM). The private blockchain platform Hyperledger Fabric refers to smart contracts as chaincodes, written in Go, node.js, or Java (Blummer et al., 2018). Independent of the programming language, usually program codes of smart contracts roughly follow the scheme “if x, then y” (Cuccuru, 2017, p. 185). Figure 17 illustrates the deployment and working of Ethereum smart dApps. To simplify the procedure, the figure does not include the illustration of the mining process.

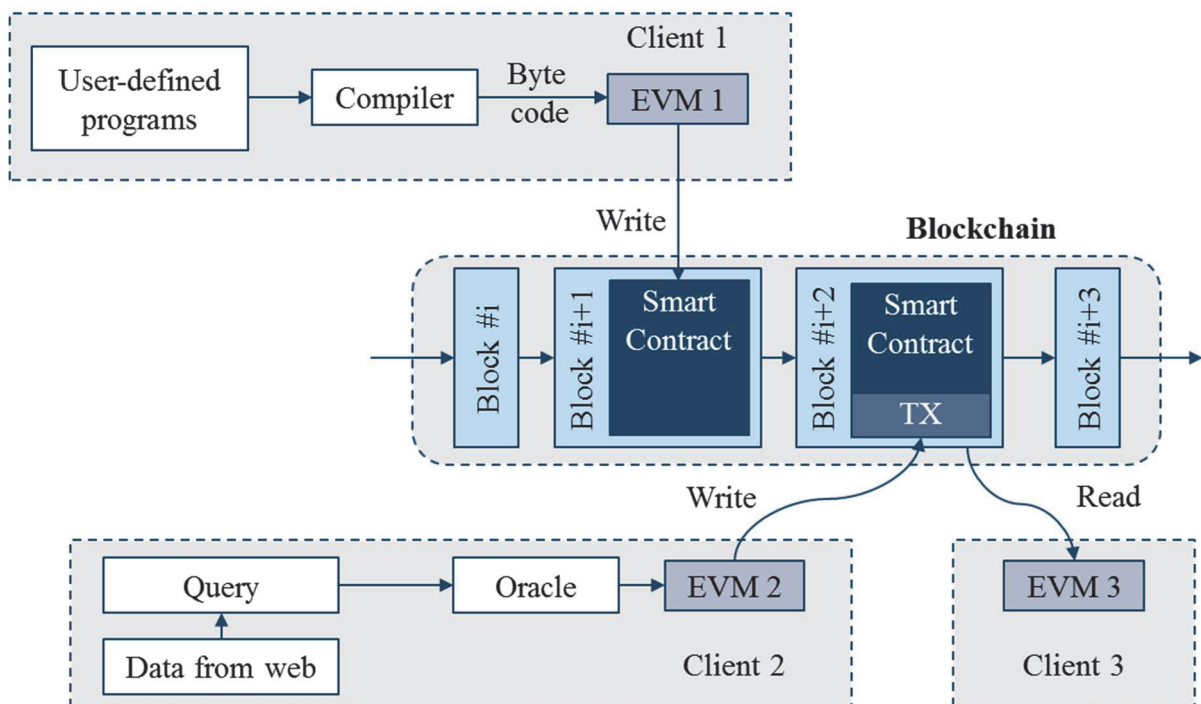


Figure 17: Mechanism of Ethereum smart contracts (based on Y. Hu et al., 2018, p. 5)

First, Client 1 creates a smart contract in a high-level language. Since this figure shows the deployment procedure of the Ethereum blockchain, Client 1 likely uses solidity for this. The smart contract is compiled into machine-level byte code where each byte represents an operation, and is then uploaded to the blockchain in the form of a transaction by EVM 1. A miner picks this transaction up and confirms it in Block #i+1. Once Client 2 has interacted via a web interface with the smart contract, the EVM 2 queries the data from the web and embeds it into the transaction TX and deploys it to the blockchain. After the confirmation of the transaction TX, the new state of the contract is updated in Block #i+2. Client 3 has to synchronise at least to Block #i+2, to see the changes caused by transaction TX.

The most extensive variation of a decentralised application is the so-called decentralised autonomous organisation (DAO). The ideal DAO is “an entity that lives on the internet and exists autonomously, but also relies heavily on hiring individuals to perform certain tasks that automation itself cannot do” (Buterin, 2014a). The most famous attempt to create a DAO on the Ethereum network was the organisation with the name ‘The DAO’. On June 17, 2016, a mistake in the complex program code of The DAO, enabled hackers to capture Ether worth of \$50 million. This attack had an enormous impact on the Ethereum blockchain and resulted in a hard fork that led to the creation of Ethereum Classic. At the same time, this case also represents the challenges and difficulties when implementing smart contracts on blockchain platforms. Once a contract is deployed on the blockchain, the immutability makes subsequent modifications and the correction of errors in the program code practically impossible (Mehtar et al., 2019; Tosovic, 2016).

2.3 Existing use cases combining distributed ledger technologies and supply chains

In this section existing use cases are presented, where the distributed ledger technology and supply chains are already combined. All presented use cases adopt the BCT. Existing supply chain applications using any other DLT technology are not known to the author at the time of writing this work. In addition, it turns out that all use cases refer only to the tracking of certain assets. Although they can change their state, they always retain their composition throughout the supply chain.

2.3.1 Blockchain technology for tracking goods

The American multinational retail corporations Walmart, and Sam’s Club, are currently working on a pilot project to improve the traceability of their leaf vegetable supply chains. Therefore, Walmart is working with IBM and 11 other food companies to develop a blockchain-based food traceability network. First results of the pilot project have shown, that retailers were able to reduce the required time to trace an item from seven days to just 2.2 seconds. The target of this implementation is on the one hand to guarantee food safety for the customers and on the other hand to reduce losses of retailers and suppliers (Blakeman, 2019).

A similar approach is currently being adopted by the Finnish retail cooperative S-Group and is named ‘the Pike-perch radar solution’. In this solution, each participant in the

permissioned platform Hyperledger Fabric, is able to record based on a blockchain, each participant records information, which can then be utilised by the other participants. By scanning a QR code on the package of the fish, the customers can trace the content back to the home water of the fish. The aim of this blockchain project is to provide the customers with more information about the food route all the way from the source to the stores and therefore increase the transparency of the food supply chain (Lehikoinen, 2018).

A further example is the tracking of diamonds by the De Beers Group, an international corporation specialised in the mining, retailing, and trading of diamonds. In this project, the BCT is used to guarantee trust by ensuring provenance, authenticity, and traceability of diamonds. The aim of the De Beers Group is to provide true traceability for the stakeholders due to the tracking of diamonds from mine to retail in order to prove that a diamond does not contain undisclosed synthetics or was involved in unethical practices (De Beers Group, 2018).

2.3.2 Blockchain technology for tracking carriers

In 2018, the German company GS1 (Global Standard One), which designs global standards to improve value chains, initiated a blockchain-based pilot project to improve the Euro-pallet exchange process. The core of this project was the pallet certificate. In the traditional process, this paper certificate belongs to the daily business of every truck driver and often causes inefficiency and lack of transparency in logistics. The certificate documents number, type and quality of the charge carried by the pallet. With the aim to digitise this process, GS1 established a decentralised blockchain network consisting of 13 external nodes belonging to different projects participants. In total, 17 participants with 20 different warehouse locations took part in order to map the pallet exchange process on the blockchain under realistic circumstances. As a result, this project proved that the blockchain is a suitable technology to improve the pallet exchange process. The blockchain not only improved the efficiency in the back office, but also simplified loading bay operations and everyday tasks (GS1 Germany, 2018).

Another example for carrier tracking by adopting the BCT is the 'TradeLens Blockchain Shipping Solution' by Maersk and IBM. The TradeLens ecosystem includes more than 20 port and terminal operators across the globe. In addition, two global container carriers, Pacific International Lines and Hamburg Sued, are participating on TradeLens. On the

blockchain, shippers, shipping lines, freight forwarders, port and terminal operators, inland transportation and customs authorities can interact efficiently and have real-time access to shipping data and documents. By the help of blockchain-based smart contracts, TradeLens enables all participants of the network to collaborate and increases the transparency for customs brokers, trusted third parties such as customs, other government agencies, and NGOs. Through better visibility, this system facilitates the container tracking process and enables the reduction of the process to find a container from 10 steps and five people to only one step and one person (Linnet & Wagner, 2018).

3 Research environment

This chapter describes the research environment and explains the constituents of the Reutlingen University and Stellenbosch University manufacturing network. Based on this environment, the chapter explains country-specific circumstances and the importance of this research for both Germany and South Africa.

3.1 The manufacturing network of Werk150

In the summer of 2019, the ESB Business School department of Reutlingen University opened a new application-oriented research fabric with the name Werk150. Werk150 offers students and companies a three-dimensional development environment with modular assembly systems, innovative conveyor technologies, collaborative robots, visual assistance systems, and modern communication technologies. Therefore, Werk150 provides an innovative infrastructure for the development and validation of application-oriented solutions in the context of Industry 4.0.

a.) Main parts of the handlebar stem



b.) Main parts of the substructure



Figure 18: a) Main parts of the handlebar stem; b) Main parts of the substructure

In this learning environment various own products are manufactured, with a scooter representing the main product. The scooter exists in five different model variants, which are almost identical in their basic construction, but differ in their colour and the materials used. Depending on the model, a scooter consists of 40 individual parts, which are assembled in 110-130 steps to form the final product. The 40 individual parts can be grouped into seven main parts, which in turn can be divided into the two main assemblies, handlebar stem and substructure. The scooter's main assemblies and their main parts are shown in Figure 18. So

far the production of scooters at Werk150, as well as the supplier network has been very local and mainly limited to Germany. In the future, more and more learning factories from partner universities will be included in the manufacturing process in order to be able to manufacture more cost-effective and individualised products. These new international manufacturing networks, however, also extend delivery routes and require new solutions for tracking the individual components. In particular, for components that are of high importance for the product's safety, the traceability of each component and its parts is of importance to clarify responsibilities and to ensure the product's quality. Therefore, traditional tracking solutions are to be expanded to ensure that all stakeholders can trace back each component and all parts it consists of at any stage of the supply chain to increase the product and network transparency. This ensures that despite the outsourcing of production processes and the involvement of international suppliers, the high quality standards for the university's own products are still met. The cooperation between Stellenbosch University and Reutlingen University will be the first step towards establishing an international manufacturing network.

The assembly consisting of footboard and rear fork is an important component of the substructure. During braking processes, high forces can act on the connection between these two parts. Since the rear wheel brake is the only integrated possibility for the user to stop the scooter, this component is important to guarantee the safety of the product. Even small inaccuracies, for example the position of the drilled holes in the footboard, can affect the safety of the scooter in the long term. The production of individual footboards may be taken over by the learning factory of Stellenbosch University in the future.

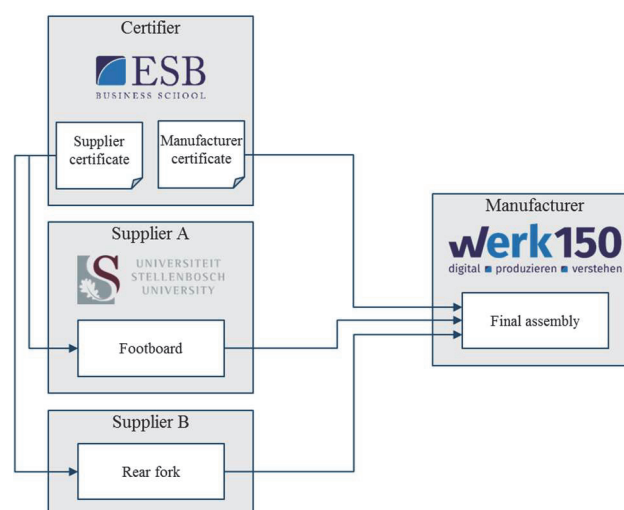


Figure 19: Assembly Structure

The assembly, whose basic structure is shown in Figure 19, consists of the two main components footboard and rear fork. Both components are produced by different suppliers, whereby the footboard will be produced at Stellenbosch University, and the rear fork will come from a German company. Only the final assembly takes place at Werk150 in Reutlingen. Since this assembly is a certified product of the ESB Business School, both suppliers must first be certificated by the ESB Business School to be able to produce the corresponding parts. In this way the quality standards can be checked and enforced at any time. This also applies to the manufacturing process itself, which must also be certified by ESB Business School. Due to the complexity of developing a solution, which in particular must include the virtual ‘merging’ of the two individual parts into a new assembly, this work will only confine itself to this simple assembly. In future projects, the findings of this work can then be transferred to the entire product portfolio of Werk150.

3.2 Country-specific circumstances

In 2018, Germany exported \$1.56T worth of goods, which makes Germany the most powerful economy in Europe, and the second largest exporting country in the world after China. Machinery including computers (\$272B), Vehicles (\$264B), and electrical machinery equipment (\$164B) represent the top three categories in terms of overall exports (Workman, 2019a). Germany’s high variety of different goods and the diverse number of trading partners, make the country the third most complex economy in the world according to the Economic Complexity Index (OEC, 2017a).

Since the early 1990s, German manufacturers and their supply chains have been undergoing a fast globalisation process. This led to an expansion to locations in new major regions and emerging economies. The globalisation of the suppliers took place at the same time, under considerable pressure from the original equipment manufacturers (OEMs). On the one hand, these require their suppliers to be present in all important world markets close to their factories and on the other hand to have a price level that can only be achieved in low-wage countries (Schwarz-Kocher, Krzywdzinski, & Korflür, 2019, pp. 10–11). This development resulted inter alia in well-known German manufacturers and suppliers such as Daimler, Volkswagen, BMW, and Bosch operating production plants in South Africa (BMW Group, 2019; Bosch, 2019; Daimler, 2019; Volkswagen, 2019).

South Africa shipped \$94B worth of products around the globe in 2017. With a total of \$7.1B, Germany is its second most important trading partner after China with \$9.2B (Workman, 2019b). In addition to the extraction and trade of precious metals, the manufacturing industry is the most significant segment of the South African economy (OEC, 2017b). The vehicle manufacturing industry, along with the automotive component manufacturing industry, represent the leading sectors of the manufacturing industry, responsible for exports amounting to a value of \$12B, representing 14,3% of the total South African exports in 2018 (Lambrecht, 2019). The South African manufacturing industry benefited greatly through globalisation, but also faces increasing pressure from challenges caused by an increasing operational complexity and increasing competition, especially against China and India (N. Lambert & Tolmay, 2017; Naude & Badenhorst-Weiss, 2011, p. 71).

Figure 20 shows a ranking of the world's most respected 'Made in' labels, whereby South Africa is ranked 38th. The label 'Made in Germany' however, is considered to be the most respected label in the world. Accordingly, South African products have only a low customer reputation, even though quality components of South African origin are involved in several prestigious German products. Car manufacturers such as BMW, Daimler, and Volkswagen even produce complete model series in South Africa (Lambrecht, 2019, p. 15). In this context, an increased transparency within the supply chain and the auditability of each product and every involved component, may be beneficial for both countries. German companies can maintain quality standards through increased transparency in complex manufacturing networks, and can increasingly include verified and eligible South African partners in their supply chains. Through the increased transparency, customers can realise to what extent components of South African origin are involved in complex manufacturing supply chains, which in turn may lead to an increase in terms of customer reputation. The exact interaction between transparency and reputation can be further analysed and validated in future studies.

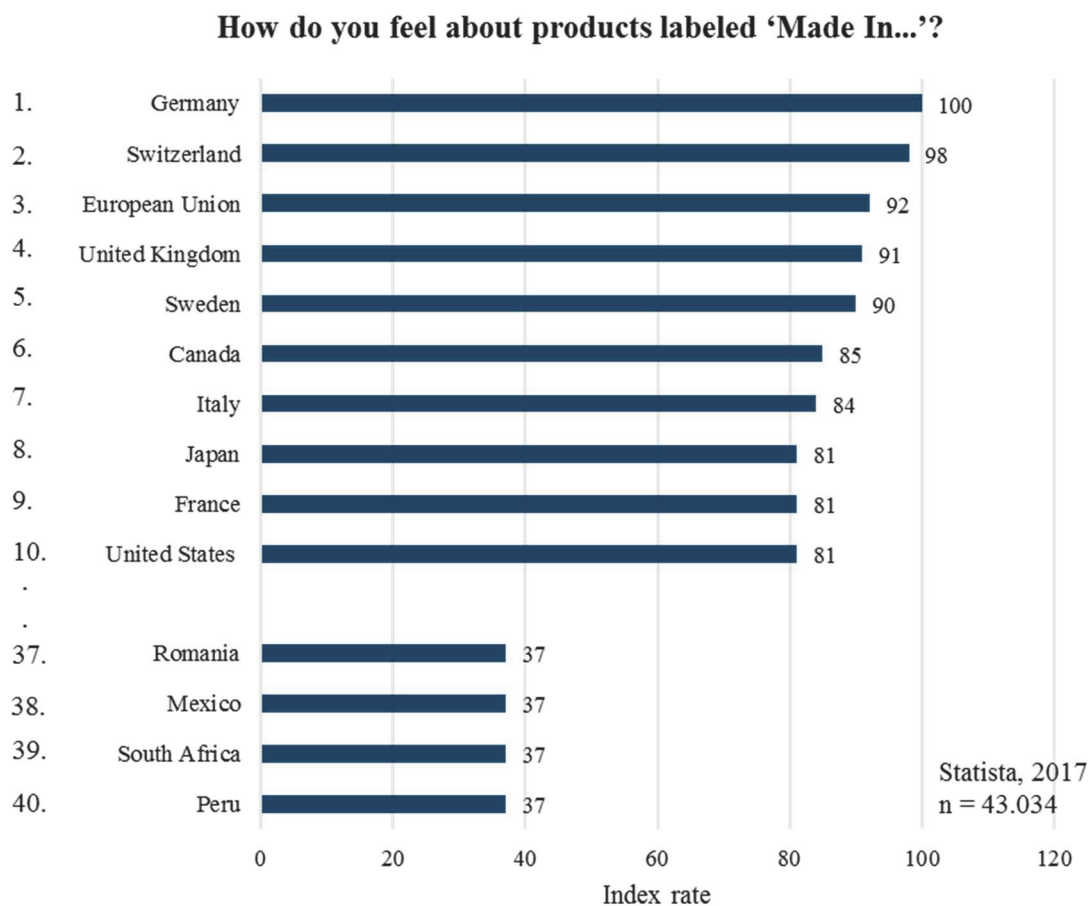


Figure 20: The world's most respected 'Made in' labels (based on Statista, 2017)

Schwarz-Kocher et al. (2019, pp. 10–11) remark that particularly in the case of German suppliers, the innovative strength of companies is based to a large extent on long, historically grown network relationships, cooperation and knowledge exchange between research and development sites in Germany, and regional universities and research institutions. The geographical closeness between production and R&D made it possible to include production knowledge in the development processes at an early stage. In this context Schwarz-Kocher et al. question whether the extent of globalisation will weaken the innovativeness of the suppliers due to the relocation of production abroad.

This research work at the Reutlingen University in Germany together with the South African Stellenbosch University aims to show that intercultural cooperation in international networks, in particular, enables the development of innovative solutions from which all sides can profit.

4 Framework development

This chapter contains the development of a framework for tracking goods in manufacturing networks by using a distributed ledger technology. First, a conceptual process is developed, based on the main stages of a SCM system framework consisting of a network design stage, network planning stage, and an execution stage (Werner, 2017). This process is then related to the properties of manufacturing networks and their implications for the properties of DLTs. Finally, a conceptual framework is proposed, combining the vertical process flow with the horizontal influences of physical and virtual complexity drivers.

4.1 Network design stage

According to the task model of supply chain management software systems (Werner, 2017, p. 87), the first main level is the supply chain design. Therefore, fundamental strategical decisions must be determined at this stage. The aim of this framework is to increase the transparency for all stakeholders and to enable traceability for a product and its components. Thus, all stakeholders directly involved in the manufacturing process must first be identified. Subsequently, all stakeholders must be assigned a network authority. The procedure shown in Figure 21 summarises the considered steps during the network design stage.

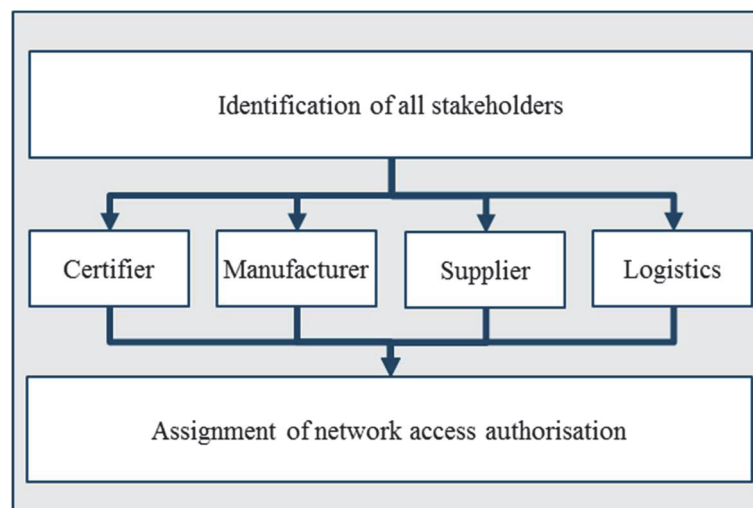


Figure 21: Considered steps during the network design stage

4.1.1 Identification of all stakeholders

As a first step, the identification of all stakeholders takes place. According to Bozarth et al. (2009), the number of suppliers in a complex supply chain and the globalisation of the supply base, can impact the complexity of this process. Companies which already have very detailed documentation of all entities involved in the manufacturing process of a product have an advantage when identifying and enlisting all participants. For this purpose, the stakeholders are divided into stakeholders directly involved in the manufacturing process and stakeholders indirectly involved in the manufacturing process.

According to S. J. Hu et al. (2008, p.46), the manufacturer, intermediate sub-assemblers, and suppliers are typical examples for entities directly involved in manufacturing processes. In the context of this work, the stakeholders directly involved in the manufacturing process are additionally expanded with the role of a certifier. Since various products can legally only be manufactured with the permission of the respective certifier, this role also has a direct influence on the manufacturing process itself. These stakeholders are then supplemented with all the entities who are indirectly involved in the manufacturing process. Typical examples for entities indirectly involved in the manufacturing process are logistics companies. The roles considered for this framework are explained in more detail below.

- 1.) *Manufacturer*. The manufacturer is seen as the final assembler in order to create the product.
- 2.) *Supplier*. The suppliers supply the manufacturer with components required to produce the final product.
- 3.) *Certifier*. The certifier provides non-physical assets such as certificates, licences, and patents necessary to legally produce the components of the suppliers and/or manufacture the final product. Depending on the characteristics of the supply chain, the role of the certifier can be taken over by the manufacturer itself, but also by other independent organisations such as governmental or non-governmental organisations.

- 4.) *Logistics companies.* Logistics are required to transport the components from the suppliers to the manufacturer or to deliver the final product to the customer. Logistics companies do not have a direct impact on the manufacturing process.
- 5.) *Customers.* Customers ultimately receive the final products and therefore represents a special peculiarity. If a high number of customers is involved in the supply chain, it will automatically result in an unknown number of network participants. They are not only unknown in terms of quantity, but also in terms of their identity. Although it requires an interface for customers in the network, they cannot be involved as active participants in the development process. However, they may be represented by organisations.

4.1.2 Assignment of network access authorisation

Once the identification of all stakeholders has been completed, they must be listed together with their respective roles. This list, together with all roles, must then be connected to the composition of DLTs and therefore with the network access authorisation. As described in Section 2.2, DLTs on a peer-to-peer basis consist of several nodes which can have different access possibilities to the network. In general, there are two main node types that can be distinguished (Reyna, Martín, Chen, Soler, & Díaz, 2018):

- 1.) *Full-node.* A node that can store the full ledger and fully validate transactions on the network.
- 2.) *Light-node.* Light-nodes do not have a complete copy of the ledger. They only keep crucial parts of the ledger sufficient for checking the validity of current transactions.

In addition, in this framework the theoretical possibility can also be considered that one or more participants of the network do not operate any node in the DLT. These participants can be given access to the network via other nodes, so that they can make transactions on the network as well. This approach could be used, for example, for logistics companies, as they may not be interested in the process of validating the transactions themselves, but still do not want to cause gaps in the tracking process. Thus they could provide their service to the network without making extensive investments in terms of a distributed network infrastructure. A node

that gives access to other entities or devices can be called a ‘supernode’ (also called overlays or gateways) in this context (Zorzo, Nunes, Lunardi, Michelin, & Kanhere, 2018, p. 3). A possible structure of such a supernode and its integration into a distributed network is shown in Figure 22.

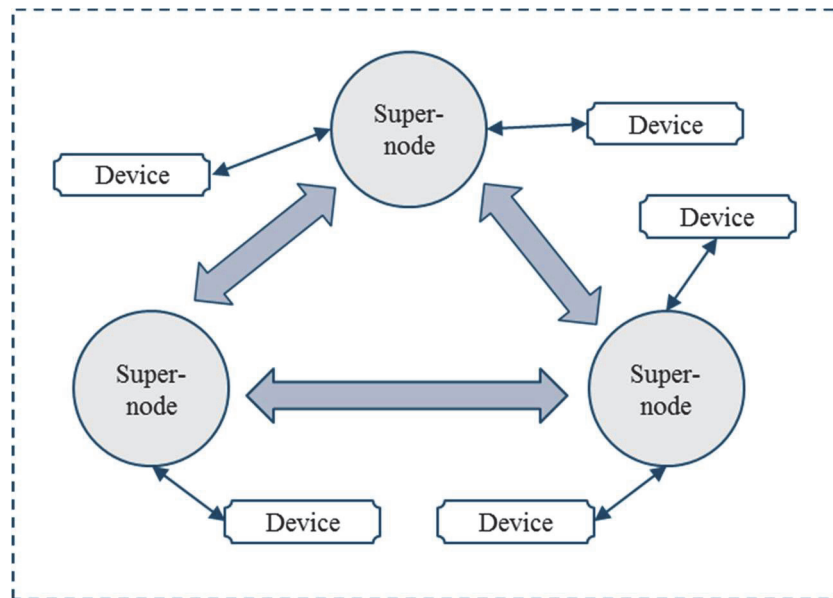


Figure 22: Peer-to-peer architecture using a supernode (based on Zorzo et al., 2018, p. 4)

As a high number of customers represents an unknown number of entities, it must be decided during the design phase whether customers are allowed to operate nodes, or only get access via supernodes. Therefore the number of customers represents a further physical complexity driver of this step. This decision, in particular, has a significant impact on the later course of this framework. Changes at any later stage, can cause a significant additional effort and therefore can greatly delay the implementation of a DLT into the SCM. For the same reason, gaps in the supply chain such as unknown suppliers, must be filled in at this phase, as they can still be fitted into the construct with relatively low effort at this point. Accordingly, every stakeholder can potentially represent a node in the network, which in turn influences the network composition. Therefore, the assignment of network access authorisation results in new virtual complexity drivers. These are characterised once by considering the number of known nodes including the customer and furthermore by considering the number of unknown nodes.

4.2 Network planning stage

The network planning is a fundamental stage when combining manufacturing networks with the use of a DLT. First, this stage analyses the impact of a product's composition on the use of DLT platform. In this case, a possibility is proposed which enables a connection between the physical world and the virtual world by creating for each asset a virtual identity on the distributed ledger. Secondly, a logic model is presented to enable a mapping of the product composition in smart contracts. Based on this logic, initial predictions can already be made about the prospective transaction per second, triggered by the manufacturing network on DLT networks. As a conclusion, this section exemplifies the impact of all previous considerations on the DLT platform decision. The procedure and all considered steps during the network planning stage are illustrated in Figure 23.

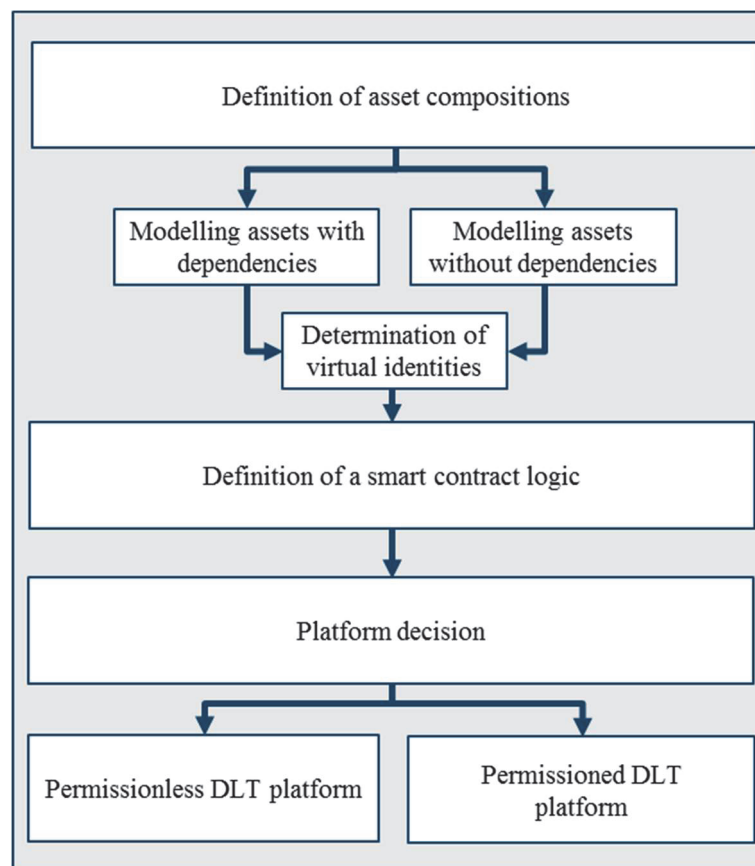


Figure 23: Considered steps during the network planning stage

4.2.1 Definition of product composition

Since the aim of this framework is to enable the auditability of a product and all its components, the entities directly involved in the manufacturing process are of great importance when defining the product composition. Figure 24 illustrates the relationship of entities directly involved in the manufacturing process based on the use case described in Chapter 3. Of course, the respective roles can be distributed arbitrarily and adapted to the needs of many different manufacturing supply chains. This figure merely serves as a foundation to illustrate the next steps of the framework planning stage.

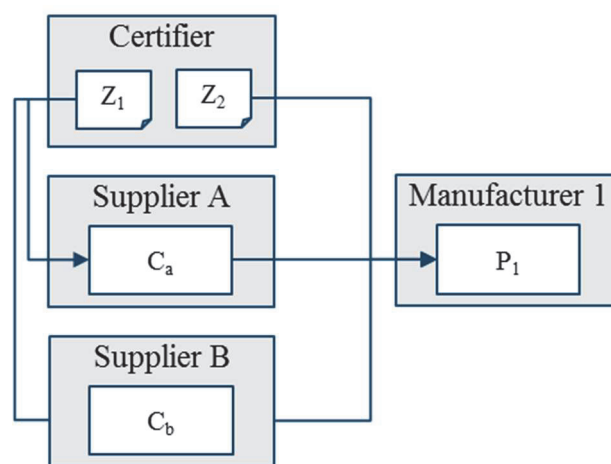


Figure 24: Relationship of all entities directly involved in the manufacturing process.

The simplified manufacturing process shown in Figure 24 involves the production of a product P_1 consisting of the two different components C_a and C_b . In order to constitute a collaborative manufacturing network, the components and the product are produced by the independent entities Supplier A, Supplier B, and Manufacturer 1. All products can only be produced when each entity owns a certain certificate Z provided by a Certifier. This exemplifies that non-physical assets such as licences, patents, or certificates can be mapped on the DLT. As mentioned in Section 4.1.1, the role of the Certifier can be taken over by the manufacturer itself, or otherwise by other independent organisations depending on the characteristics of the respective supply chain. The relationships between the assets and entities shows that the complexity of a product impacts the complexity of the definition the most. In this context, the term complexity refers to the number of individual parts of a product and entities involved in the manufacturing process. In addition, the batch size also influences the complexity of the production processes (Bozarth et al., 2009) and therefore complicates the definition of a

product's composition. A product which is manufactured in many different and individual variations, can differ considerably in its product composition and the entities involved. Thus a concrete determination of the composition is more complicated than with products which are produced in high and constant lot sizes.

Once all relations of the product and its components have been defined, they must be transferred to the logical structure of DLTs. Grossman (2015) describes DLTs such as the BCT as “public database of timestamps”. In traditional models, each system kept its own notion of time. In DLTs however, all nodes of the network agree on a common time by implementing a consensus algorithm. As a result, the immutability of the network guarantees that a certain event on the network happened at a certain time. For illustrative purposes, this comparison between the traditional model and the DLT-based model is shown in Figure 25.

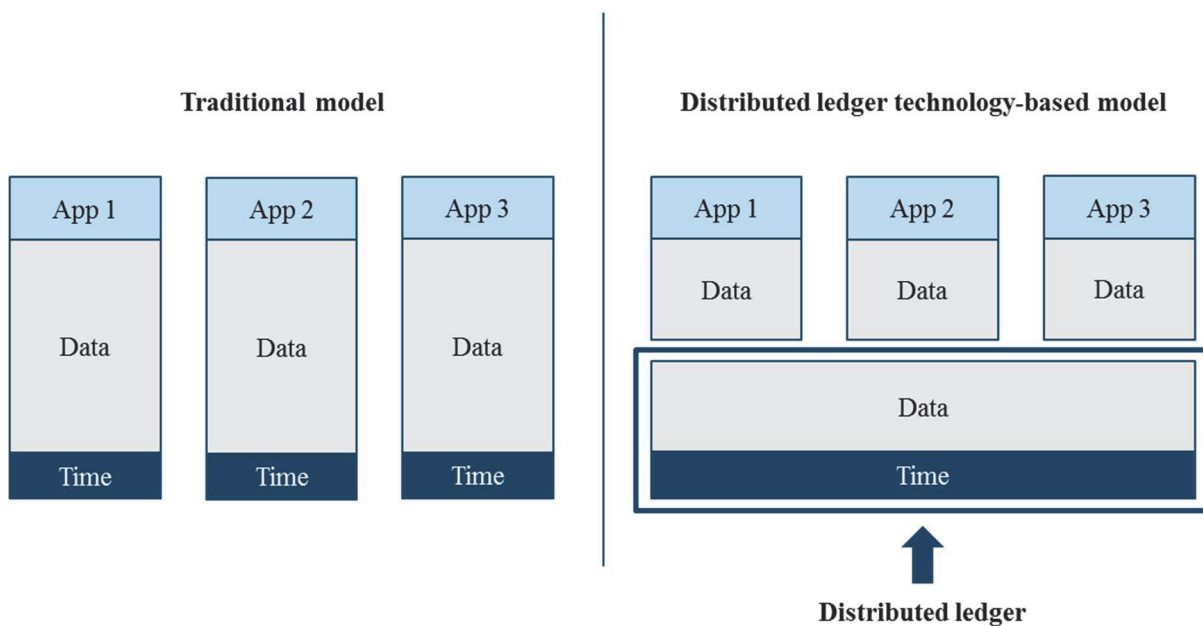


Figure 25: Distributed ledger technologies as verified public timestamps (based on Grossman, 2015)

Based on this insight, the immutability of DLTs also requires the products and components of manufacturing supply chains to be structured in a temporal sequence. Essentially, all requirements of the physical supply chain must be transferred to the virtual supply chain. In the physical world, the time relationships between processes are usually very clear. For example it is not possible to assemble a certain product without having the required parts on hand. This relationship must exactly be considered when connecting all components logically

on the distributed ledger. In principle, a distinction can be made between two different types of assets.

- 1.) *Assets without dependencies.* Those are assets that can be created without depending on previous actions. Assets without dependencies could be for example raw materials or certificates. The number of assets without dependencies depends largely on the characteristics of the respective manufacturing supply chain. In case of the model of the manufacturing process (Figure 24), this type of asset is only represented by the certificates, since the creation of certificates does not depend on any previous asset or action.
- 2.) *Assets with dependencies.* Those are assets that can only be created when previous actions have been successfully conducted. In case of the model manufacturing process for example, the creation of the components C_a and C_b depends on the receipt of the respective certificates Z_1 . The manufacturing of P_1 even requires the receipt of the components C_a , C_b and a manufacturing certificate Z_2 . Again, the number of assets with dependencies depends greatly on the characteristics of the respective manufacturing supply chain.

An essential aspect of the product composition is also to determine in which way and to what extent the involved assets are mapped on the DLT. As mentioned in Section 2.1.4.2, the information sharing among supply chain members can be divided into the sharing of product information, transaction information and inventory information (Khan & Yu, 2019, pp. 27–29). To enable sharing of information on a DLT, the physical asset's properties must first be logically mapped to the distributed ledger. Such a logical link on a decentralised network is considered as decentralised application (dApp). Buterin (2013) differs between financial, semi-financial, and non-financial applications. Accordingly, a manufacturing process can be considered as a non-financial application since it does not include any financial aspects. If the manufacturing process were additionally linked with financial transactions, this would result in semi-financial applications. Since this framework, however, deals with the problem of components that are grouped into assemblies or products, the non-financial aspect of manufacturing supply chains is paramount to this work. This characterisation of manufacturing supply chains thus requires a logic that allows traceability of assets and their

status changes such as the merging of components to assemblies, and at the same time it requires a tracking of ownership changes of these assets. In order to solve this issue, this work proposes the creation of virtual identities representing the physical assets on the DLT.

4.2.1.1 Determination of virtual identities

To ensure traceability by using a DLT, a link between the DLT network and the physical asset must be established. Therefore this framework proposes a system whereby each asset is represented by a unique virtual identity on the distributed ledger. Accordingly, this approach assumes that all virtual identities can refer to an asset itself. Above all, the uniqueness of each number is of great importance. Only in this way can it be ensured that any asset is unmistakably represented by its virtual identity on the network, which then enables the auditability of this asset.

With hashing algorithms, most DLTs such as the BCT already have the ability to generate unique hexadecimal numbers embedded in the technology (Section 2.2.2.1). For this reason, this approach uses these hashing algorithms in order to create the unique identification numbers (IDs) to connect physical assets with their virtual identities. These IDs are generated by the respective algorithm of the DLT by hashing the information they refer to. In this work, numbers resulting from such a procedure are therefore called Hash IDs. In the case of a product of a manufacturing supply chain, for example, the product information can be used to create the Hash ID of the product. The extent of information used for this process depends on the network design and is closely linked to the objectives of the respective network. If the focus is on increasing the visibility of the supply chain, this information can contain detailed information for use within the manufacturing supply chain. If, however, the focus is more on increasing transparency, it makes more sense to use information relevant for the customer.

As mentioned in Section 2.2.2.1, hash functions use typically deterministic algorithms, which means that the same input always creates the same output (Brennan & Lunn, 2016, pp. 19–20; Dang, 2012, p. 4). This leads to the fact, that hashing algorithms alone do not make Hash IDs unique. For example, the hashing of the same product information always results in the same Hash ID, which counteracts a differentiation of products of the same type. Consequently, the product information must be supplemented with a variable input, which can guarantee uniqueness when creating Hash IDs. To guarantee uniqueness, this approach

proposes to involve the timestamp function of DLTs at the moment of creating new Hash IDs, since the integrity of timestamps in DLTs makes a reactive change impossible (Lee & Deng, 2018, p. 168). The product information together with the timestamp is thus ‘condensed’ to a unique hexadecimal hash.

Hash IDs work similarly to primary keys in traditional relational central databases and are unique identifiers for a set of information. A double assignment of primary keys in traditional relational databases is obviated by the central system itself. Additionally, the primary keys are chosen by the system or the user without any direct connection to the set of data they refer to (Rolland, 2003). Hash IDs however, are generated by an algorithm by hashing the information they refer to. Therefore, Hash IDs are a logical result of their input data and provide initial information about the origin, composition, and time of creation of the asset they refer to. Accordingly, Hash IDs can also be referred to as new and decentralised versions of primary keys. A comparison between the characteristics of traditional primary keys and Hash IDs is shown in Table 7.

Table 7: Comparison between primary keys in relational central databases and Hash IDs

Identifier type	Primary key	Hash ID
Technology	Relational central database	Distributed ledger technology (DLT)
Creation	Algorithm of central database	Hashing algorithm of DLT
Reference	Refers to a set of information	Refers to a set of information
Traceability	No logical connection between primary key and set of information	Logical result of the information the Hash ID refers to
Uniqueness	Only unique within the central system	Uniqueness guaranteed by the algorithm in the entire distributed network

Within the distributed ledger, these Hash IDs represent a unique virtual identity of their physical counterparts. In order to map a manufacturing process, the unique virtual identities must have the same ownership and conversion characteristics as their physical counterparts. In this aspect they differ fundamentally from conventional primary keys in central systems. Therefore, virtual identities must always be clearly assigned to an owner and must be able to change their owners, for example, when the physical product is sold. In this context, smart contract-based Hash IDs have combine the characteristics of asset-backed tokens (Section 2.2.2) with the characteristics of conventional central database systems.

Furthermore, the virtual identities must be able to be summarised, for example when combining individual components to a new product. Since virtual identities can represent assets with dependencies and assets without dependencies, their creation on the distributed ledger must be linked to logical dependencies. In order to map such a logic, it requires the modelling, planning, and definition of further various smart contract functions.

4.2.2 Definition of a smart contract logic

When all processes in the supply chain are logically linked and the time relationships are created, this logic must then be reflected on the DLT together with the possibility to move, merge, and trace the created virtual identities. The so-called smart contracts are the essential component for mapping the logic of the supply chain and to enable the creation of virtual identities. Even the term ‘smart contract’ as coined by Buterin (2014) with blockchain-based applications, this term is used in this work as a generic term for program codes in order to program decentralised apps (dApp) of any DLT. A typical dApp consists of smart contracts on the distributed network, user interface, and backend resources such as off-network storage (Dhillon et al., 2017, p. 42). In order to enable a programming of such smart contracts, basic models of the logical relationships must first be created. To do this, this step links the functions of the smart contracts of each asset to the entity which is responsible for its initial creation. This means that only the entity which ‘owns’ the process of creating a certain asset in the physical world, is able to have access to the smart contract to create the virtual identity of the part. For this purpose, addresses are created on the distributed ledger for each entity, which are then linked to the respective functions of the smart contracts. In the case of BCT, a connection is established between the public key and the smart contracts on the blockchain. Only the owner of the appropriate private key is then able to trigger the respective actions of the smart contract (Dhillon et al., 2017, p. 42).

First, the logic model is described using the assets without dependencies described in the previous section. The model shown in Figure 26 enables the creation of virtual identities for assets without dependencies. When the possession of the smart contract is clearly assigned, information for creating a new virtual identity can be entered into the user interface. In case the proposed Hash IDs are used, the information stored behind each Hash ID at the time of creation, can for example come from conventional central systems owned by the entity creating the Hash ID. This also creates an interface between conventional systems and the

blockchain. As described by Dhillon et al. (2017, p. 42), triggering smart contract functions always results in transactions on the distributed network. At the moment of this transaction, the timestamp of the network is added to this transaction to guarantee the uniqueness of each Hash ID. The new state of the smart contract, together with the owner's address of the virtual identity is then stored on the network.

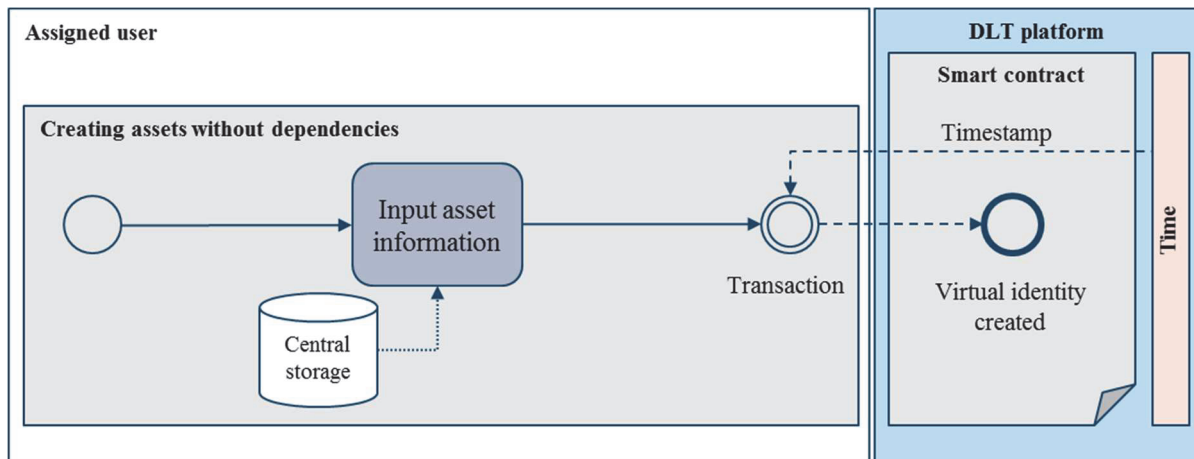


Figure 26: Model to create virtual identities for assets without dependencies

Assets with dependencies are based on the fulfilment of predefined conditions. Before a user is able to create a new virtual identity of this type, the smart contract executes a query function to check whether all conditions for creating the new virtual identity are fulfilled. In manufacturing networks for example, this query function checks the possession of all required components for producing a product. After the query function has been executed successfully, the information on the asset will be added as for the assets without dependencies. At the moment of the transaction trigger, the virtual identity required for creation are additionally marked as 'used up', since the DLT does not allow the deletion of information. This prevents assets from being used more than once to create new virtual identities. The model to create virtual identities for assets with dependencies is shown in Figure 27.

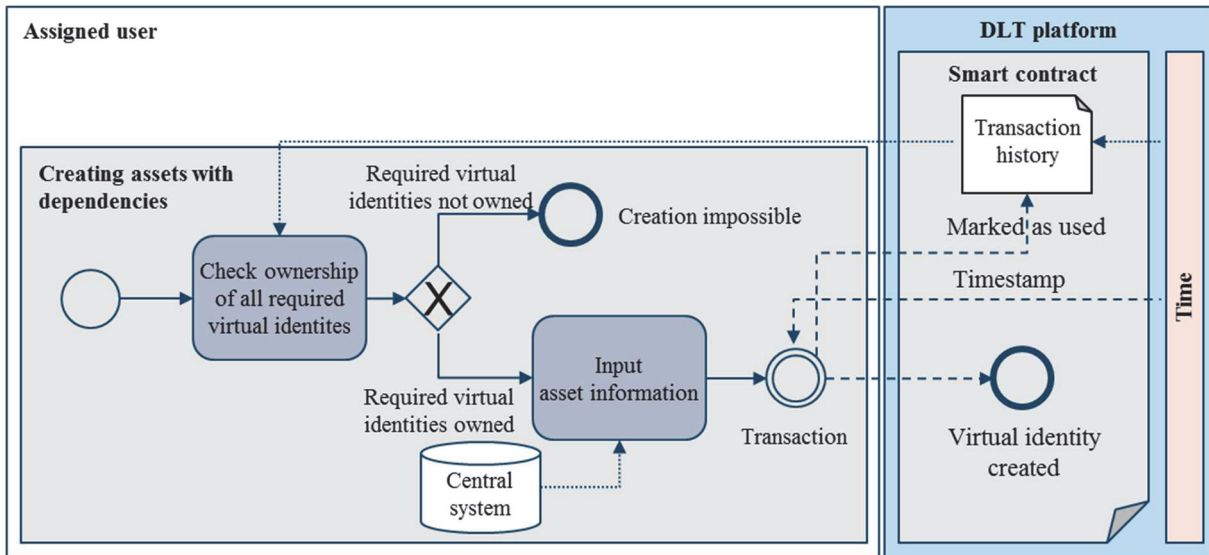


Figure 27: Model to create virtual identities for assets with dependencies

The creation of assets with and without dependencies must additionally be supplemented with a model allowing ownership changes of each virtual identity. The creation of assets is firmly bound to the address of the responsible entity. This means that only a selected authorised entity is able to create the respective virtual identity. The sending function, however, is variably accessible to the current owner of the asset. After creating a new virtual identity, it is first in the possession of the creator itself. The address of the creator is then authorised to send this virtual identity to a new owner. As soon as the resulting transaction has been successfully confirmed by the network, the creator no longer has the rights to cause further actions related to the sent virtual identity. Only the address of the recipient can trigger further actions regarding the received virtual identity. As with the creation of assets with dependencies, the sending process is therefore preceded by a query of ownership. The model to send virtual identity of assets is presented in Figure 28.

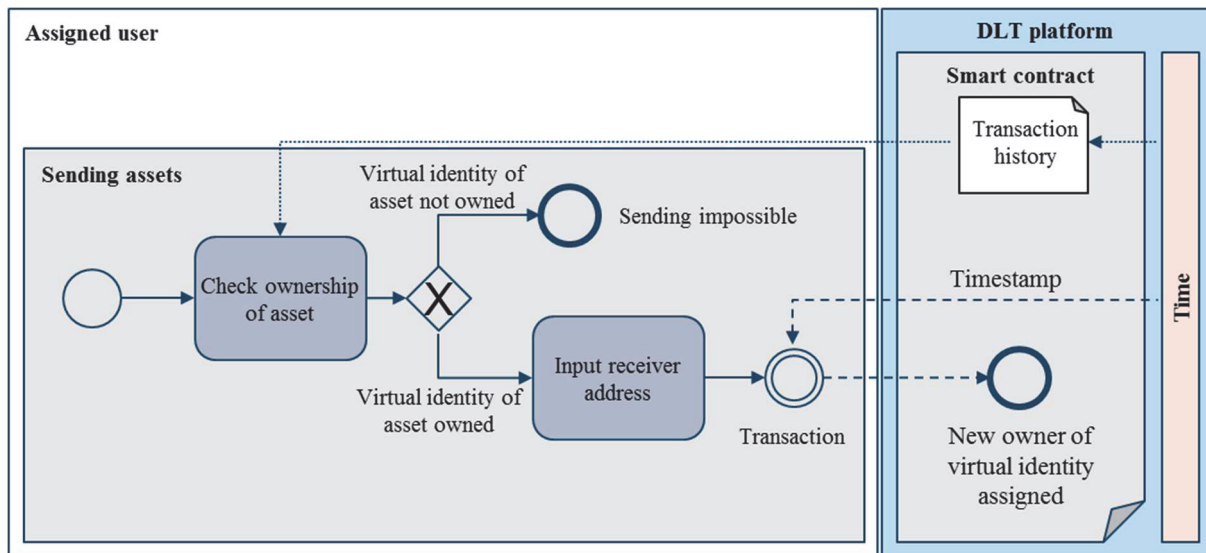


Figure 28: Model to send virtual identities of assets

In manufacturing supply chains which include, for example, modular assembly processes, an alternating sequence of creation and transmission processes takes place (S. J. Hu et al., 2008, p. 46). Figure 29 shows the smart contract models structured in an alternating sequence. This sequence first includes the creation of a new virtual identity representing an asset without dependencies. This identity is then sent to a new owner, which in turn allows the new owners to create a new virtual identity representing an asset with dependencies. After the creation, this virtual identity can be sent to a new owner as well. The horizontal arrangement of such alternating sequence of actions as shown in Figure 29, results in a very similar structure as described by Grossman (2015) and presented in Figure 25 on page 57. All actions are immutably recorded in their chronological order by the DLT. Even though central systems are linked to the DLT, it represents only a static connection. This means that only at the time when a transaction on the DLT is triggered, an image of the central system's status at the time of the transaction is created on the distributed network. This record is invariable according to the characteristics of the DLT. Therefore, changes of data records in the central systems can only be taken into account in all transactions taking place after the change has been made.

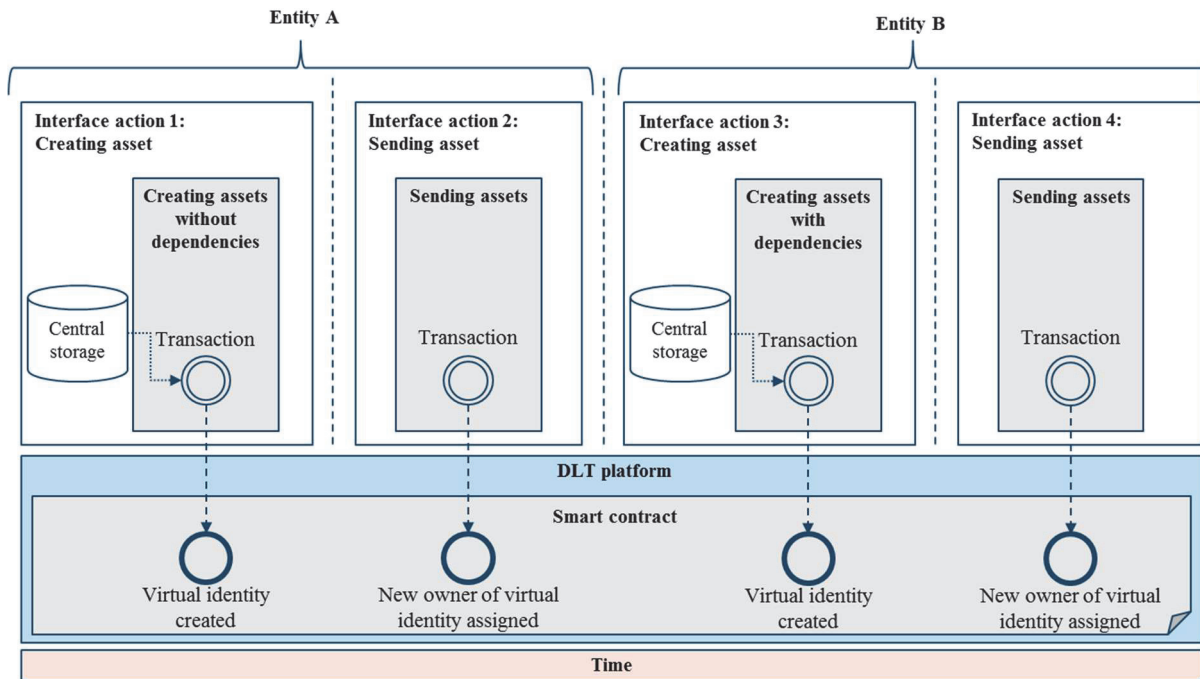


Figure 29: Smart contract models arranged in an alternating sequence

Figure 29 illustrates that every action triggers a transaction on the distributed ledger. When transferring this to manufacturing supply chains, every change of state of a product or component results in a transaction on the distributed ledger. This affects the change of state in terms of ownership as well as the change of state in terms of product composition. Based on this, it is possible to predict an initial number of caused transactions.

4.2.2.1 Prediction of transactions per second

As mentioned in the previous section, the number of transactions caused in order to produce one product depends on the number of state changes and ownership changes. In this context, state changes refer to the changes of a product, for example in production processes, where two components are assembled to create one new product. When transferring this knowledge to manufacturing supply chains, it shows that products consisting of large numbers of components and parts, cause more transactions than products consisting of few parts and components. At the same time, the ownership changes of the product impact the transaction rate. This relationship between changes of state and changes of ownership and their impact on the number of transactions is illustrated in Figure 30.

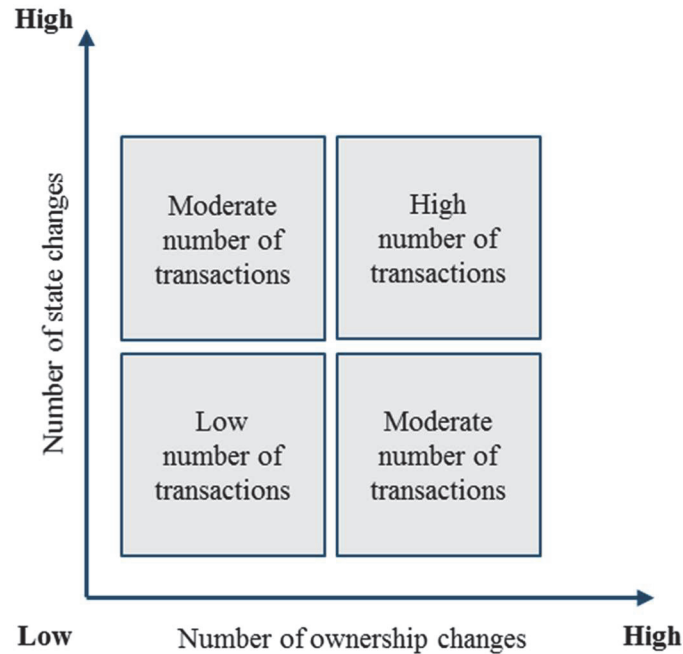


Figure 30: Correlation between manufacturing processes and total number of transactions

The total number of ownership changes depends on the one hand on the total number of entities involved to produce a certain product, and on the other, the design of the tracking process. Figure 31 shows two different designs of the tracking process. Process (a) shows a tracking process where each entity owns only one address. In such a process, it is only possible to know that an asset is currently owned by a certain entity. In process (b), Entity A owns three addresses representing incoming warehouse, production, and dispatch warehouse. In such a design, the tracking process is more detailed and allows access to more detailed information about the current state of the product.

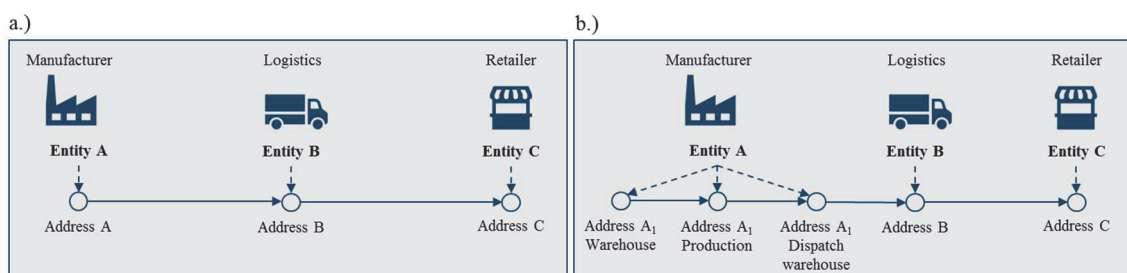


Figure 31: a.) Tracking process with each entity owing one address; b.) Detailed tracking process where Entity A owns many addresses

The number of total transactions caused by one product, is therefore the sum all its changes of state s_n and all its changes of ownership o_n . Additionally, it is not only important to calculate

the total number of transactions, but also in which period of time these transactions take place. This does not only apply to one production process, but to all parallel concurrent processes taking place in the same time interval Δt . Equation (1) below shows this mathematical relationship between a number of n concurrent processes in a certain time interval to calculate the predicted transactions per second *PTPS*.

PTPS : Predicted transactions per second

s_n : Changes of state of asset _{n}

o_n : Changes of ownership of asset _{n}

Δt : Time interval in seconds

$$PTPS = \frac{s_1 + o_1}{\Delta t} + \frac{s_2 + o_2}{\Delta t} + \dots + \frac{s_n + o_n}{\Delta t} = \frac{\sum_{i=1}^n (s_i + o_i)}{\Delta t} \quad (1)$$

The calculated prediction is an important component needed for the platform decision. In particular, critical intervals must be checked with small time intervals in order to identify bottlenecks with a high number of transactions per second. Accordingly, the predicted number of transactions per second results in a virtual complexity driver, which influences the process in the later course of the conceptual sequence.

4.2.3 Platform decision

As the final step of the planning phase, a suitable platform for implementation must be found. Due to the properties of DLTs, different technologies can be considered, since the properties required for the implementation of this framework are not technology specific, but strongly platform dependent. For example Bitcoin and Ethereum are both based on the BCT. Nevertheless, the two platforms differ significantly (Vujicic, Jagodic, & Randic, 2018). Therefore, the decision of where to implement the models and findings of the previous sections, depends more on the platform and its characteristics than on the DLT.

The term ‘platform’ in the context of DLTs, is not clearly defined in literature. Software platforms are generally described as “operating environments upon which applications can execute and which provide reusable capabilities such as file systems and security” (Bottcher, 2018). Related to DLTs, platforms define basic properties such as

consensus algorithm, data and transaction format, support of smart contracts, and network composition. These properties have a significant impact on the scalability and network size of the platform (Pahl, Ioini, & Helmer, 2018, p. 5).

The findings of the previous sections are used as a prespecified foundation for the platform decision and are therefore summarised below:

- 1.) *All relationships can be logically mapped* (Section 4.2.). All relationships of the manufacturing supply chain can be logically mapped and connections can be created. This affects ownership relationships as well as the logical dependencies of the product or component composition.
- 2.) *Data refer to an asset itself* (Section 4.2.1.1). The data can refer to an asset itself in order to enable an identification of each asset. Therefore, a virtual identity representing the asset on the distributed ledger is necessary. This framework proposes the creation of a unique identifier in the form of Hash IDs as a possible solution.
- 3.) *Data history can be logically linked* (Section 4.2.2.). The data history can be logically linked and put in a chronological sequence. This enables a transparent tracking and auditability of each asset's history.

To map such extensive relationships and complex logical links on a DLT platform requires a platform that supports Turing Completeness. Turing Completeness is a mathematical concept and is a measure of the computability of a programming language. Therefore, the language design includes complex constructs such as loops and conditions which enable the creation of all types of general purpose programs (Lee & Deng, 2018, p. 155). In particular, the categorisation of manufacturing supply chains as non-financial applications, requires extensive programming possibilities in any aspect and not from a financial application point of view. These programming possibilities must be available to the smart contracts on the platform, which are needed to create, send and merge virtual identities. In case the proposed Hash IDs are used to create virtual identities, the smart contracts also require the inclusion of hashing algorithms and timestamps. This first part of the platform decision flow is illustrated in Figure 32.

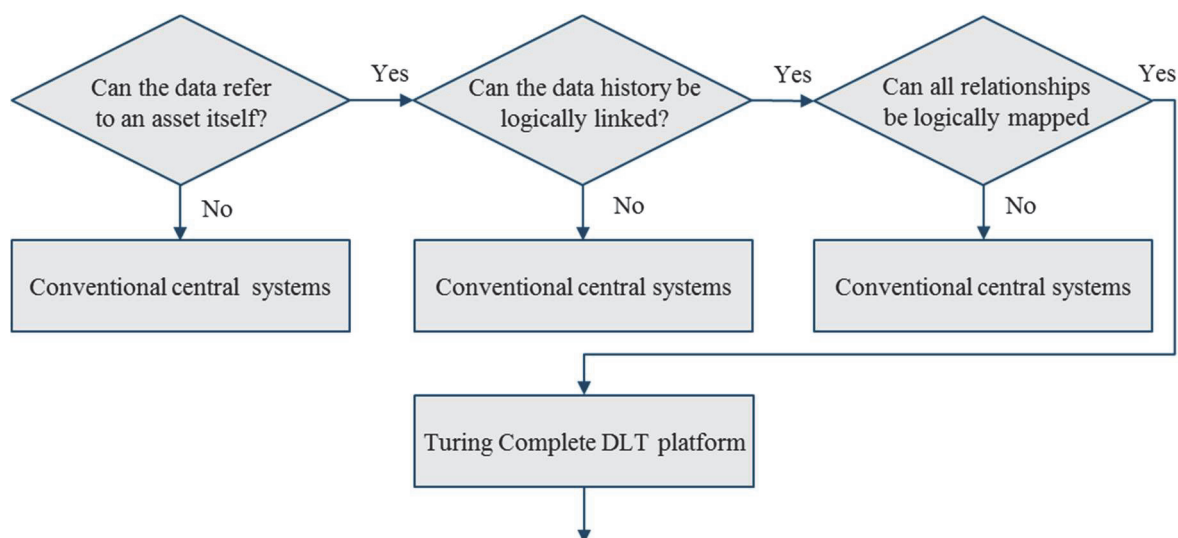


Figure 32: Part 1 of the platform decision flow

4.2.3.1 *Permissionless or permissioned platform*

One main aspect of the platform decision is the decision between a permissionless and a permissioned DLT platform. Essential prerequisites for this decision were already defined during the design phase. Basically, the decision depends on whether at the time of network creation and in the future, all network participants can be regarded as known entities. In this context, especially the network access categorisation of customers described in Section 4.1.1 is significant. According to Section 4.1.1 a high number of customers involved in the manufacturing supply chain always results in an unknown number of network participants. In case it is requested to give the customers full access to the platform and the ability to operate nodes on the network, only a permissionless network is feasible. Conversely, restricted customer access, for example via supernodes, can only take place on permissioned platforms.

According to an analysis by Iansiti and Lakhani (2017), the adoption of foundational technologies typically happens in four phases. Each phase is defined by the novelty of the application and the complexity of coordination efforts needed to make them workable. Applications low in novelty and complexity gain acceptance first, while it can take decades to evolve applications high in novelty and complexity. As shown in Figure 33, Iansiti and Lakhani (2017) categorise self-execution smart contracts, such as those used in order to map supply chain scenarios, as high in both complexity and novelty.

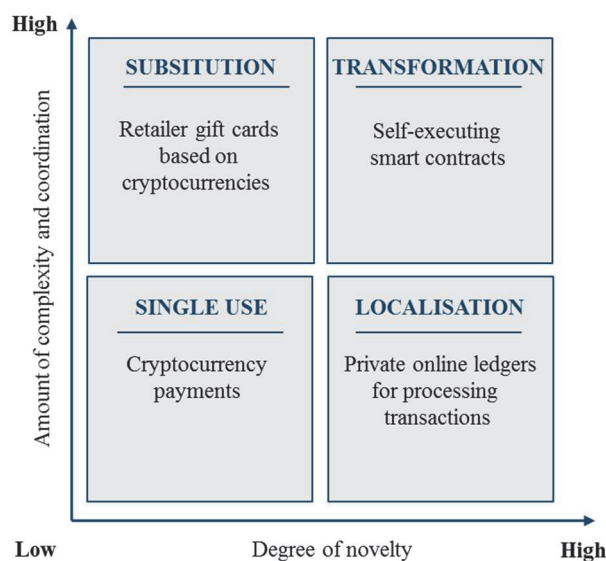


Figure 33: Adoption of foundational technologies (Iansiti & Lakhani, 2017, p. 7)

According to this analysis, it is also possible to start in on a private platform before opening the platform to the public in the future, mainly because manufacturing supply chains are very complex networks high in novelty. In such an approach, however, it should be noted that not every platform is capable of being used both as a permissioned and a permissionless platform. The concrete aspects which need to be considered are explained in more detail in the next chapter. Additionally, the choice between permissioned and permissionless platforms is not only related to the factors mentioned above. The scalability of DLTs currently represents a bottleneck when implementing decentralised applications (Zheng et al., 2017). Therefore, scalability and the relationship with the properties of manufacturing supply chains are considered in more detail in the following chapter.

4.2.3.1.1 Scalability analysis

The scalability represents a current limitation of DLTs. In this context the term scalability refers mainly to the number of transactions a platform can process per second (Zheng et al., 2017). These scalability issues result in an inherent bottleneck of DLTs, in particular of current blockchain-based platforms (Mueller, 2019). Table 8 exemplifies the scalability differences of DLT platforms.

As shown in Table 8, the permissionless blockchain platform Ethereum is only able to process 15-25 TPS (Njui, 2018). The permissionless platform EOS, however, can scale almost up to 4,000 TPS (Fadilpasic, 2019). The permissioned blockchain Hyperledger Fabrics can

reach a transaction throughput of at least around 3,500 TPS (IBM Research Editorial Staff, 2018). Also the DLT platform IOTA based on the tangle-technology only reached 183 TPS in 2017 (Schiener, 2017). NANO, a platform based on the Block-lattice architecture technology reaches 306 TPS (Pugh, 2018). As a comparison, Visa conducted a stress test in 2013 where the central system reached 47,000 TPS (Trillo, 2013).

Table 8: Scalability comparison between DLT platforms

Platform	Technology	Type	Transactions per second (TPS)	Source
Ethereum	Blockchain	Permissionless	25	Njui, 2018
EOS	Blockchain	Permissionless	4,000	Fadilpasic, 2019
Hyperledger Fabrics	Blockchain	Permissioned	3,500	IBM Research Editorial Staff, 2018
IOTA	Tangle	Permissionless	183	Schiener, 2017
NANO	Block-lattice architecture	Permissionless	306	Pugh, 2018

The table also shows, that although Ethereum and EOS, for example, are based on the same technology, they still show significant differences in terms of their scalability. A closer comparison between the two platforms shows that they have a different design in terms of decentralisation and security. On the Ethereum platform, the majority of the entire network must agree to the validation of transactions (Buterin, 2013). On the EOS platform however, new transaction blocks are produced by a selected group of only 21 block producers (EOS Authority, 2019). This relationship between decentralisation, scalability and security is also known as the scalability trilemma. This trilemma refers to the current impossibility to create a system that is at the same time highly decentralised, secure, and scalable (Viswanathan & Shah, 2018). This correlation is shown in Figure 34.

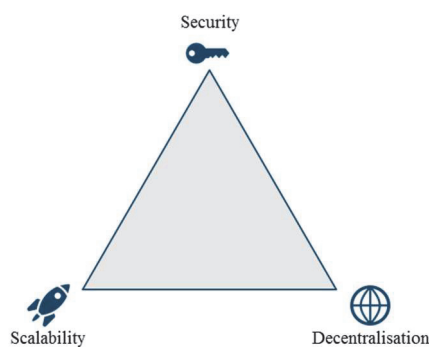


Figure 34: The scalability trilemma (based on Viswanathan & Shah, 2018)

The required number of transactions depends individually on the respective manufacturing supply chains. The calculation conducted in Section 4.2.2.1, provides an initial indication of how scalable the platform must be. In particular, the consensus mechanisms of DLT platforms are the bottleneck of scalability issues (David Im, p. 14). In this context, permissionless platforms with low scalability on public networks should not be excluded in advance for the use as permissioned networks. Consensus mechanisms that exist on permissionless networks can always be used for permissioned networks as well. Consensus mechanisms which were created especially for permissioned networks, however, cannot insist on public permissionless networks (Dinh et al., 2017, p. 1088). This is especially important because, as described by (Iansiti & Lakhani, 2017), a slow development from private networks to public networks can be considered as a realistic scenario. To avoid misunderstandings, the next section analyses the difference in the properties of permissionless consensus mechanisms used in a permissioned environment.

Permissionless public DLT platforms, especially of the BCT, are often associated with a low scalability (Chauhan, Malviya, Verma, & Mor, 2018; Zheng et al., 2017) and high energy consumption (Truby, 2018). In addition, there are false statements caused by a lack of knowledge about DLTs, which could make companies insecure to use permissionless platforms in a permissioned environment. For example Baumann and Supe (2018, p. 14) state, that with increasing length of the blockchain, for example with an increasing number of transactions, the degree of difficulty of the blockchain calculations increases drastically. As a result, more computing power, energy and time is required to complete the transactions, which is neither sustainable nor suitable for the current use in SCM.

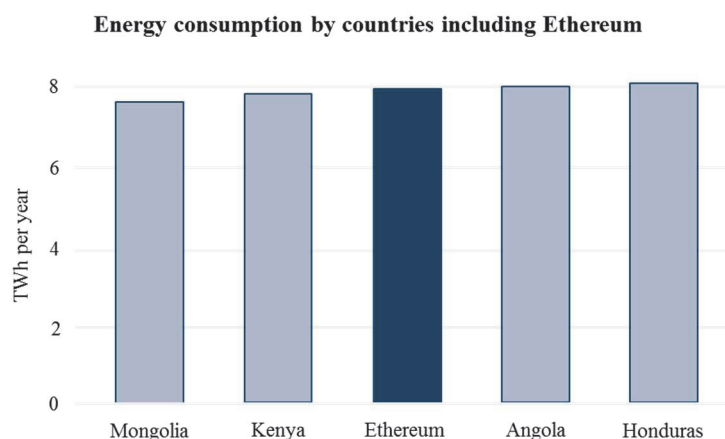


Figure 35: Energy consumption by country including Ethereum (digiconomist, 2019)

The chart shown in Figure 35 shows the energy consumption by country, including the Ethereum main network. This illustrates that Ethereum requires as much energy as small countries in order to maintain the stability of the network. However, the energy consumption is not in any way related to the number of transactions as stated by Baumann and Supe (2018, p. 14). Therefore, a short analysis of the PoW algorithm is intended to eliminate ambiguity in terms of energy consumption in order to facilitate the platform decision for companies and the use of DLTs in the SCM. Even though the PoW algorithm is mainly a blockchain-based algorithm, it clarifies the key element of DLTs in terms of energy consumption and is representative of other algorithms and their differences between a permissionless and permissioned network. The necessary background to fully understand the following simplified scenario is provided in Section 2.2.2.3.

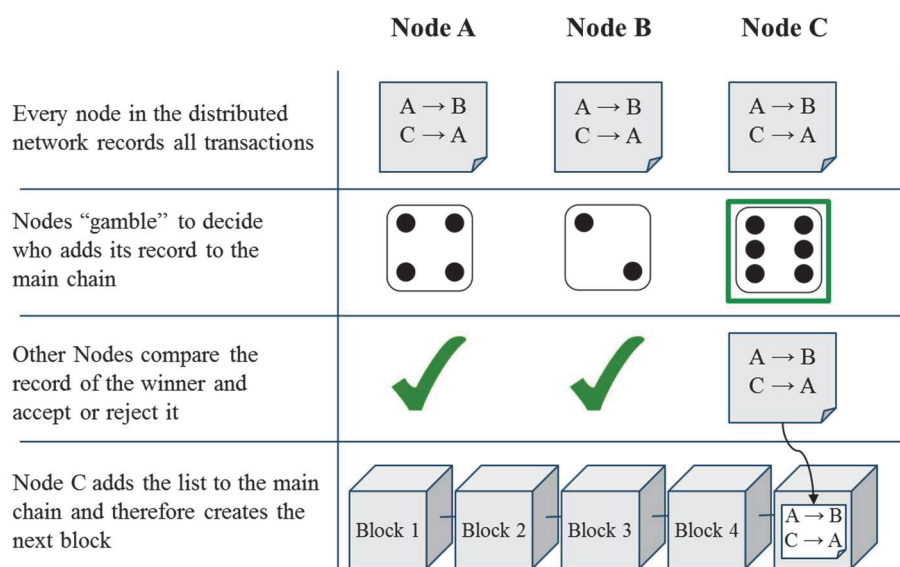


Figure 36: Simplified proof-of-work in a permissioned network with a fixed difficulty and a fixed number of participants

In a simplified PoW scenario it is assumed that a BCT requires a fixed average block time in order to validate transactions correctly. Therefore, the platform defines a so-called difficulty, which statistically guarantees that this average block time is adhered to by the network (Wood, 2014). A simplified procedure to validate transactions and create new blocks, is presented in Figure 36. The PoW algorithm is thereby simplified illustrated by nodes which resemble a rolling dice. The node which is rolling a six first, can add the next block to the chain. This dice game represents the required computation to solve the arithmetic problem, which is normally carried out by nodes on PoW-based platforms (Baliga, 2017, p. 6; Mingxiao et al., 2017,

p. 2568; Nguyen & Kim, 2018, p. 106). Since Figure 36 represents a network with a fixed number of participants, the difficulty can also be determined as a fixed value. Thus, the energy consumption required to 'roll the dice' can be considered as constant.

On a permissionless network another picture emerges. By opening the network to the public, any number of participants can join the network and participate in the 'dice game'. Due to the fact that now a higher number of participants 'rolls the dice' at the same time, it statistically decreases the required time until one participant successfully throws a six. However, as described above, the platform needs a fixed time to confirm transactions without any errors. In order to counterbalance this fixed time, the platform sets the difficulty to a variable value. In the figurative sense, the difficulty of the game is increased until the specified block time is reached again. This effect is further enhanced when the winners of the game are financially rewarded in public networks (Baliga, 2017, p. 6; Mingxiao et al., 2017, p. 2568). As a result, nodes join the network, which specifically try to increase their chances by using more computing power. Conversely, this again increases the difficulty of the platform. This effect can swing up until the profit is no longer in equilibrium with the costs required for the energy consumption or hardware and therefore the validation of blocks is considered uneconomical.

This comparison illustrates that the energy consumption is not related in any way to the chain length. It is an interaction between a large number of network users that use a high amount of computing power and a platform which has a variable difficulty in order to keep the block time constantly on the same level. In a permissioned network, this interaction can be counteracted by adapting the required computing power to the number of network participants. Accordingly, PoW-based platforms can also be considered in the platform decision for permissioned networks without the disadvantage of high energy consumption, but with the advantages in terms of security and decentralisation.

Although the transactions per second supported by the platform cannot be increased when using an originally permissionless platform as a permissioned platform, the whole capacity of the network can be used for one's own application. In case of a permissionless network, the application of the manufacturing supply chain has to share the capacity with other network participants or applications.

The holistic platform decision process is illustrated in the platform decision flow shown in Figure 37. As illustrated in the decision flow, all virtual complexity drivers have an impact on the platform decision itself. Since in this work supply chain transparency refers to the disclosure of information to all stakeholders, only the results of the decision flow including the customer are further considered during the practical implementation and validation of the framework.

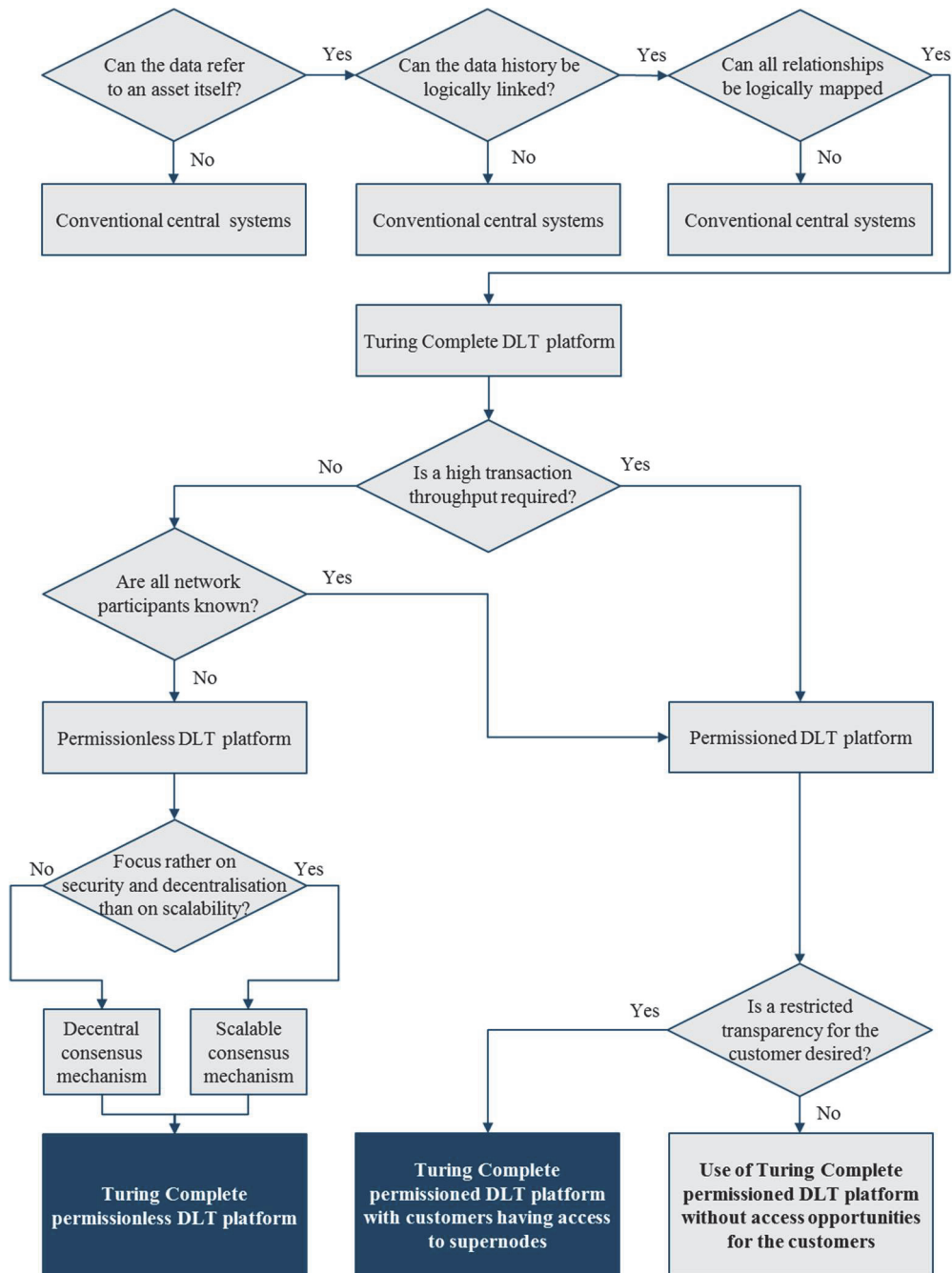


Figure 37: Platform decision flow

4.3 Execution stage

During this stage the practical implementation and the vertical and horizontal integration of the platform takes place. The supply chain and all its logical interrelationships must be completely mapped in the blockchain network in order to ensure a complete traceability, authenticity, and auditability of each product and its components. This happens first on a software level, whereby the selected platform is integrated into the manufacturing network. Subsequently, all supply chain processes must be adapted to the new network and components such as the Hash ID must be firmly integrated into the production processes. In doing so, this approach does not differ between permissionless and permissioned networks. However, the result and the degree of transparency for the customer vary. The steps which are considered during the execution phase are shown in Figure 38.

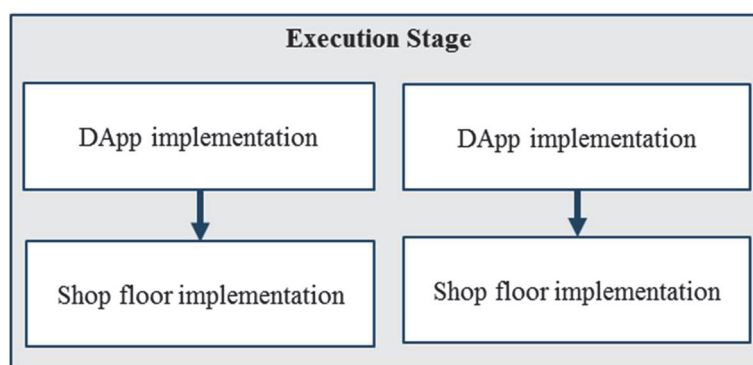


Figure 38: Considered steps during the execution stage

4.3.1 DApp implementation

During the implementation of the decentralised application, the transfer of the created logic model to the selected DLT platform takes place. There are no differences between permissionless and permissioned network types when transferring the logic model. By applying the smart contract logic described in Section 4.2.2, the chronological sequence of the physical process can be mapped virtually on the DLT. The implementation takes place in the respective programming language of the platform. The Ethereum platform, for example, uses ‘Solidity’, its own object-oriented programming language to create smart contracts on the EVM. The permissioned platform Hyperledger Fabric refers to smart contracts as so called ‘chaincodes’, which can be created in the languages Go, node.js, or Java (Blummer et al., 2018). Since the Turing Completeness is a component of the platform decision process, only

platforms that can reliably implement the programming of smart contracts are considered for this step. Additionally, each smart contract needs a suitable user interface, which allows the users to interact with the code. Even though the program code differs depending on the programming language, the logic of the code remains the same. As an example, the format of the Hash IDs depends on the embedded hashing algorithms of the platform. The format of Hash IDs is more or less similar, since most of the hashing algorithms used by DLTs are based on the SHA-256 algorithm, resulting in 64 hexadecimal characters (Dang, 2012, pp. 6–7). For example Ethereum smart contracts use the Keccak-256 algorithm, which also results in 64 hexadecimal characters, only differing in that they always begin with a ‘0x’ (Wood, 2019).

The new DLT network can be integrated into the supply chain network software environment. Depending on the extent of the DLT use, the DLT network can either supplement existing tracking systems or even replace them completely. As described in Section 4.2.1.1, central systems can also be integrated as part of the platform to store important information about a certain asset at the moment of creating its digital identity. In particular, the main actors of the supply chain network must provide the computing power to run the respective decentralised platform. Additionally, an address represents each supply chain participant on the network. Furthermore, each participant needs an account to trigger transactions. This account is also referred to as a wallet, consisting of a public key and private key (Bruehl, 2017, p. 136). These addresses linked to the smart contracts allow the participants to create inputs via their user interface. In a permissionless network any user is able to create a pair of public and private keys. In a permissioned network, on the other hand, the creation of a key pair can only be done by a user operating a node. If all these conditions are met, the creation and sending of virtual identities on the network can be performed and tested.

Figure 39 shows a permissionless platform implementation scheme according to the model of the manufacturing supply chain. First, the certifier is creating certificate Hash IDs and assigning them to the addresses of the suppliers and the manufacturer. The introduction of a number of ‘n’ suppliers in this scheme, elucidates the scalability and the possibility of continuous extension of this scheme. After receiving the certificate, the suppliers are able to create their respective components and submit them to the manufacturer. The manufacturer is only able to create the virtual identity of product 1, if the manufacturer’s account owns all the required Hash IDs of all components and certificates. All Hash IDs can change ownership as

often as desired and, for example, be assigned to the addresses of different distributors, retailers or in case of the end product, also to the customer. Due to the immutability of the DLT, tracing the history of each virtual asset is possible at any point of the supply chain. Since Figure 39 represents a permissionless network, the customer is able to take part in the process as a full node. Therefore, customers are able to receive and own parts on the network and to have them at their disposal. As illustrated, the customer does not have any access to a smart contract and is therefore not able to actively take part in production processes and change the product's composition. It still allows customers to be the verified owner of Hash IDs on the network, representing the virtual identity of a physical product.

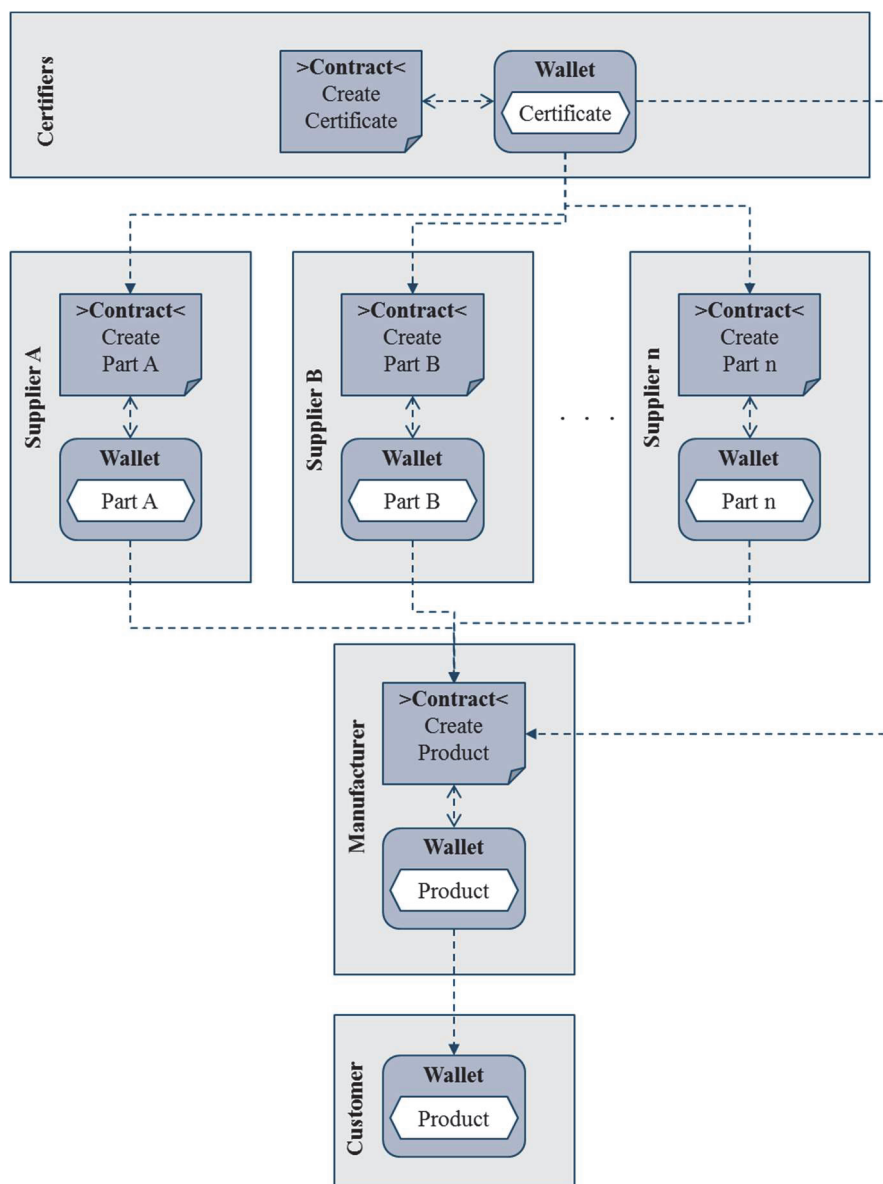


Figure 39: Permissionless platform implementation scheme

In contrast, Figure 40 shows the implementation scheme of a permissioned platform. The entire process to produce a product is the same as on the permissionless network. However, the customer has no opportunity to join the network as a node. In this case, the customers only get access to a supernode, which is represented by the manufacturer in this example. This access can, for example, take place via an internet application. The differences between the two implementation schemes, particularly in terms of customer transparency, must be verified and validated based on a practical implementation.

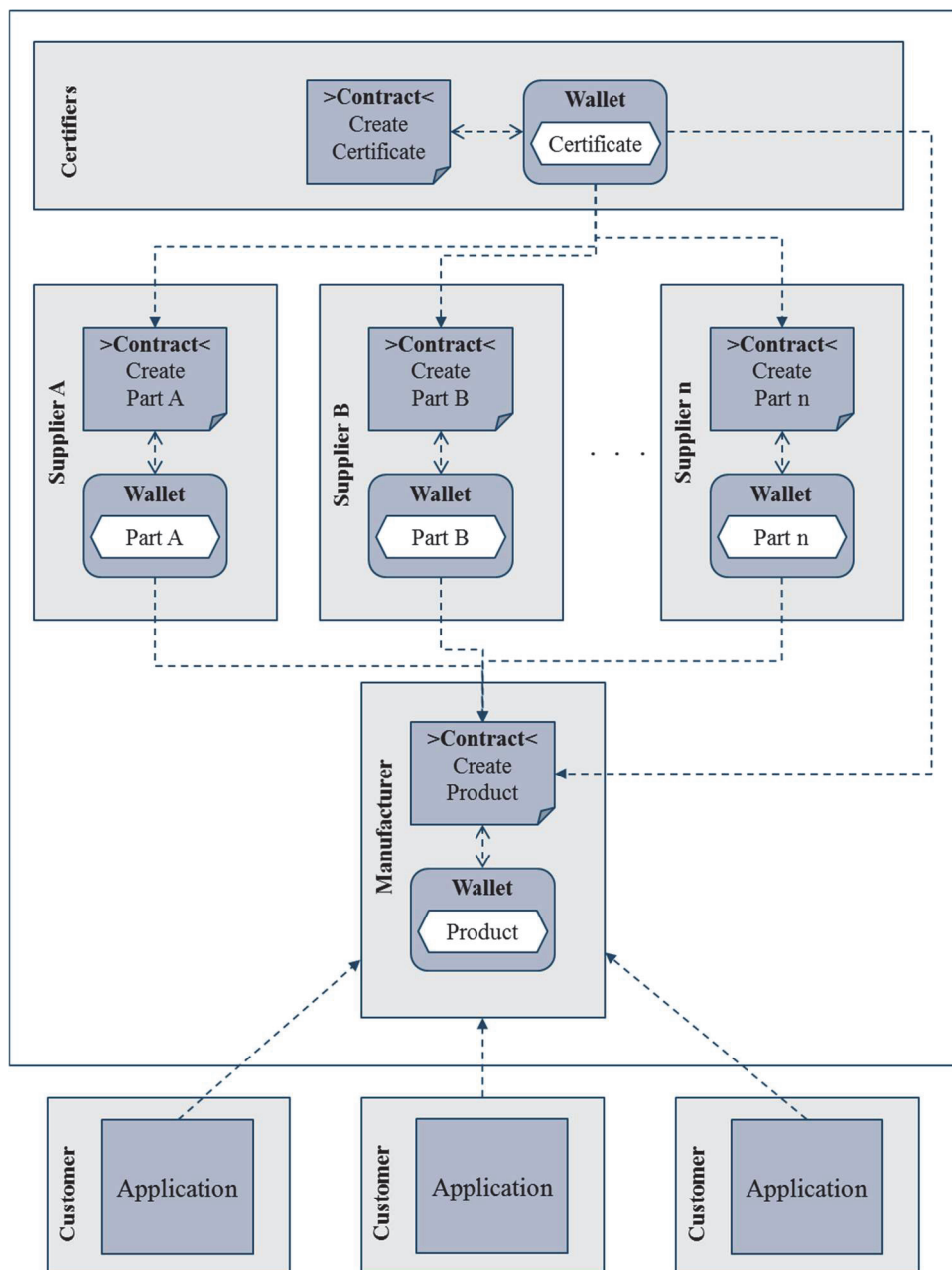


Figure 40: Permissioned platform implementation scheme

4.3.2 Shop floor implementation

As a final step, the network structure must be linked to the physical processes and integrated into the shop floor environment. This particularly concerns the embedding of Hash IDs into the production process. Hash IDs represent the virtual identity of an asset on the distributed network. Therefore, the Hash IDs must be attached to the asset itself. Section 2.1.4.4 already introduces the common use of the barcode technology and the RFID technology in tracking systems. Both technologies are promising solutions for attaching Hash IDs to physical components or products. Which technology is better suited, depends mainly on the nature of the asset itself. It is important, above all, that the attached tags can survive the entire life cycle of the respective asset. Only then can a product and its components be traced back to their origins even years after production.

- 1.) *Barcode technology.* Barcodes or QR represent a cheap and easy method to tag products with Hash ID. They are easy to create and are characterised by their longevity. Depending on the product they, the size of the code can be adapted. However, too small codes cannot be read well by reading devices. This makes this solution particularly unsuitable for tiny components.

- 2.) *RFID technology.* When adopting the RFID technology only the use of passive RFID chips for attaching Hash IDs makes sense to consider, since active RFID chips do not meet the aspect of longevity. Hash IDs usually are of 256-bit lengths, which represents a very small data string (Dang, 2012, pp. 6–7). This enables the use of micro RFID chips to tag tiny products or components. RFID chips also offer advantages in the identification of components which are assembled in such a way that there is no visual contact with the reading device.

Hash IDs remain unchanged throughout their entire lifetime. This also allows you to consider engraving this number onto the components or products. This can also be done in the course of an adaptive manufacturing process. Thus the product is already provided with the Hash ID during the production process and linked to its virtual identity. This approach can be particularly advantageous for assembly processes carried out by humans. The Hash ID can be recognised with the naked eye and the hands remain free, because no reader has to be used.

It is not necessary to commit to one approach or technology for tagging the physical assets. The different approaches and technologies can also be mixed within the supply chain. The most important aspect is the coordination of the involved participants to enable the planning and standardising of production processes. For example, the appropriate reading devices must be integrated into the respective production processes to ensure that the right components are always selected. In order to guarantee complete traceability of a product and its components, it is no longer possible to assemble any arbitrary components. It must be ensured that the same components that are assembled on the DLT platform are also assembled in the physical world. For this to be possible, components must be clearly identified before each production step. Due to the immunity of DLTs, the decentralised application does not allow retroactive changes.

4.4 Proposed framework

The framework combines a vertical conceptual SCM system implementation process with horizontal influencing complexity drivers in order to enable the tracking of manufactured goods and all components they consist of by adopting a DLT. This results in a conceptual framework consisting of a network design stage, a network planning stage, and an execution stage.

In the network design stage, the framework suggests to identify and categorise all stakeholders involved in the manufacturing supply chain. As Figure 41 shows, the complexity of the identification process depends on the globalisation of the supply base and the number of stakeholders. The identified stakeholders are categorised according to their roles within the supply chain, to which the respective network authorities are subsequently assigned. As stated in the course of the framework development process, a high number of customers represents an unknown number of entities, which in turn results in an unknown number of network nodes. Therefore, this stage results in two virtual complexity drivers; the number of nodes and the number of unknown nodes. Finally, the stage results in an initial design of the network's structure and all its associated nodes and their respective authorities.

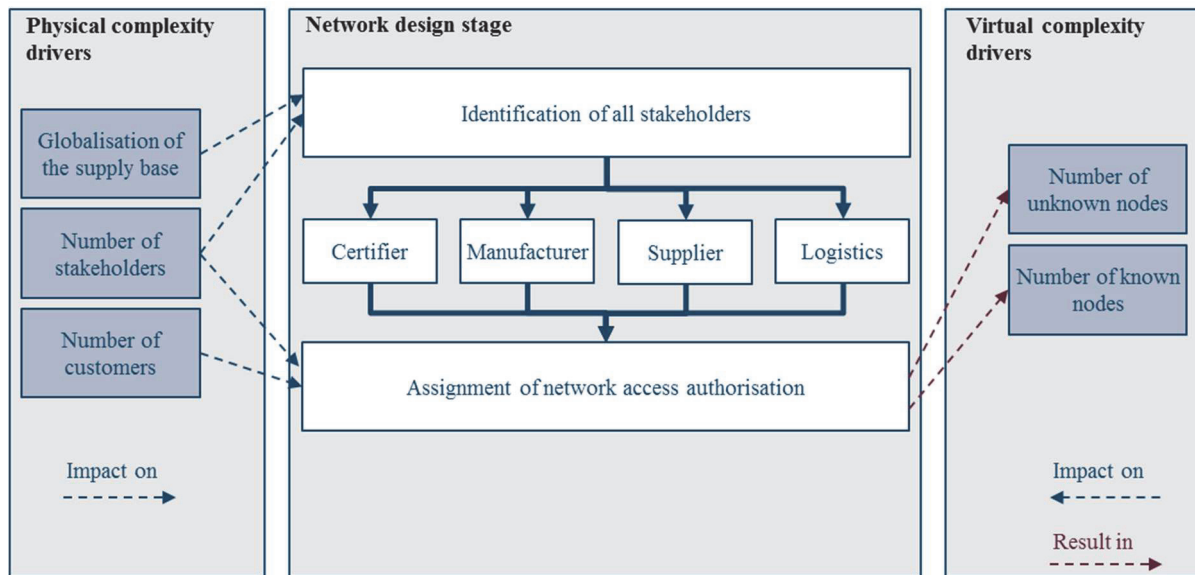


Figure 41: Complexity drivers influencing the network design stage

Based on the network design, the framework recommends defining the respective asset compositions in the network planning stage. The complexity of the products and the batch sizes influence the complexity of these steps. The definition of the asset composition results in logical dependencies between the respective assets, which must be mapped virtually. Therefore, with the logical dependencies a further virtual complexity driver is formed. Fundamentally, the framework distinguishes between assets with dependencies and assets without dependencies. In order to track and trace a product by using a distributed ledger technology, each asset must obtain a virtual identity on the distributed ledger which must have the same ownership and conversion characteristics as their physical counterparts. To meet these requirements the author developed the approach of smart contract-based Hash IDs. The framework proposes to define a smart contract logic consisting of models, which determine the dependencies for creating and linking the virtual identities logically. Depending on these models, it is possible to analyse the number of state changes and ownership changes of each asset. Based on this analysis and the number of concurrent processes, it is possible to predict the number of transactions per second. The complexity of the platform decisions is impacted by all virtual complexity drivers, since a platform must be able to meet all these requirements. The complex logical dependencies require the platform to support a Turing Complete programming language to enable a mapping of these on a distributed ledger. The scalability of distributed ledger technology platforms is currently a limiting factor for decentralised applications with high transaction density. Additionally, the platform and in particular its consensus mechanism must therefore have sufficient capacity in terms of scalability to support

complex supply chain applications. Depending on the number of known nodes and the number of unknown nodes, the platform decision can result in a permissionless platform or a permissioned platform.

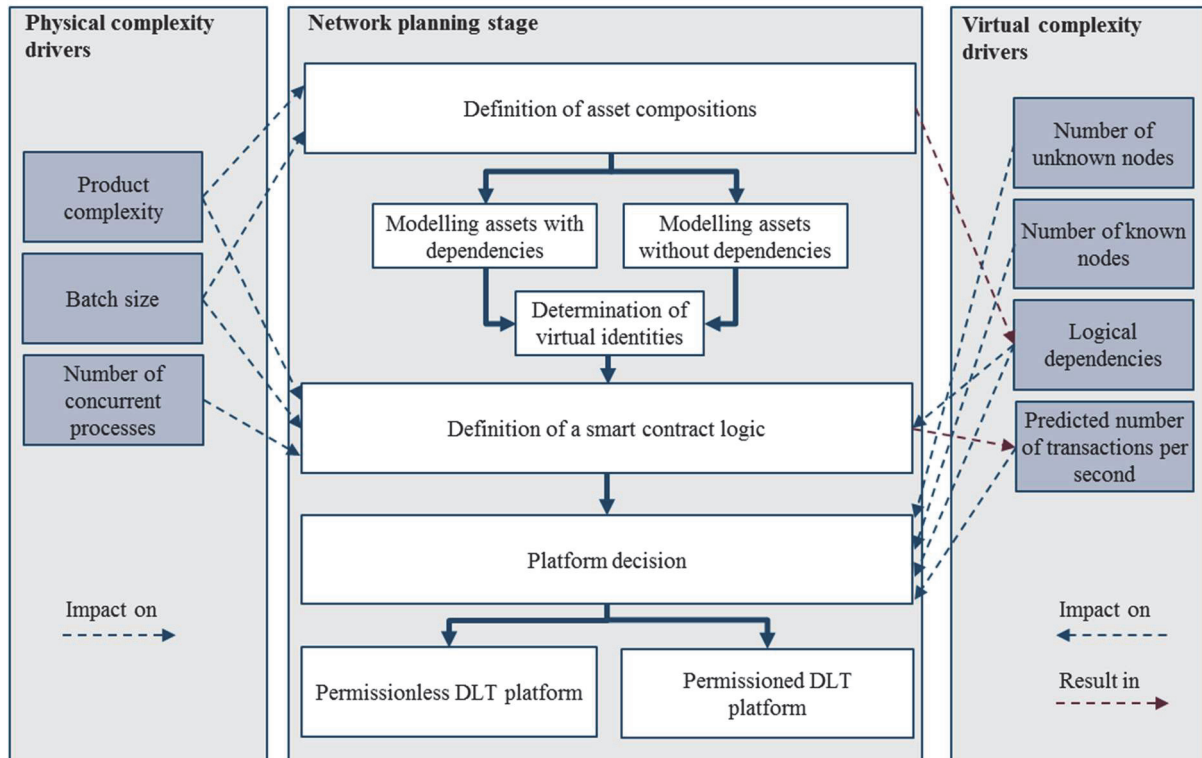


Figure 42: Complexity drivers influencing the network planning stage

In the final stage, the execution stage, the framework describes the holistic implementation and integration of the smart contract-based decentralised application. Therefore, the smart contract logic models are implemented on the selected distributed ledger technology platform and can subsequently be integrated into all production and logistics processes. The implementation process itself does not differ between using a permissionless or permissioned network. However, it influences the provided customer transparency. The exact impact on the customer transparency needs to be verified and validated based on a practical implementation. Figure 43 shows the holistic conceptual framework for tracking goods in manufacturing networks by using a distributed ledger technology.

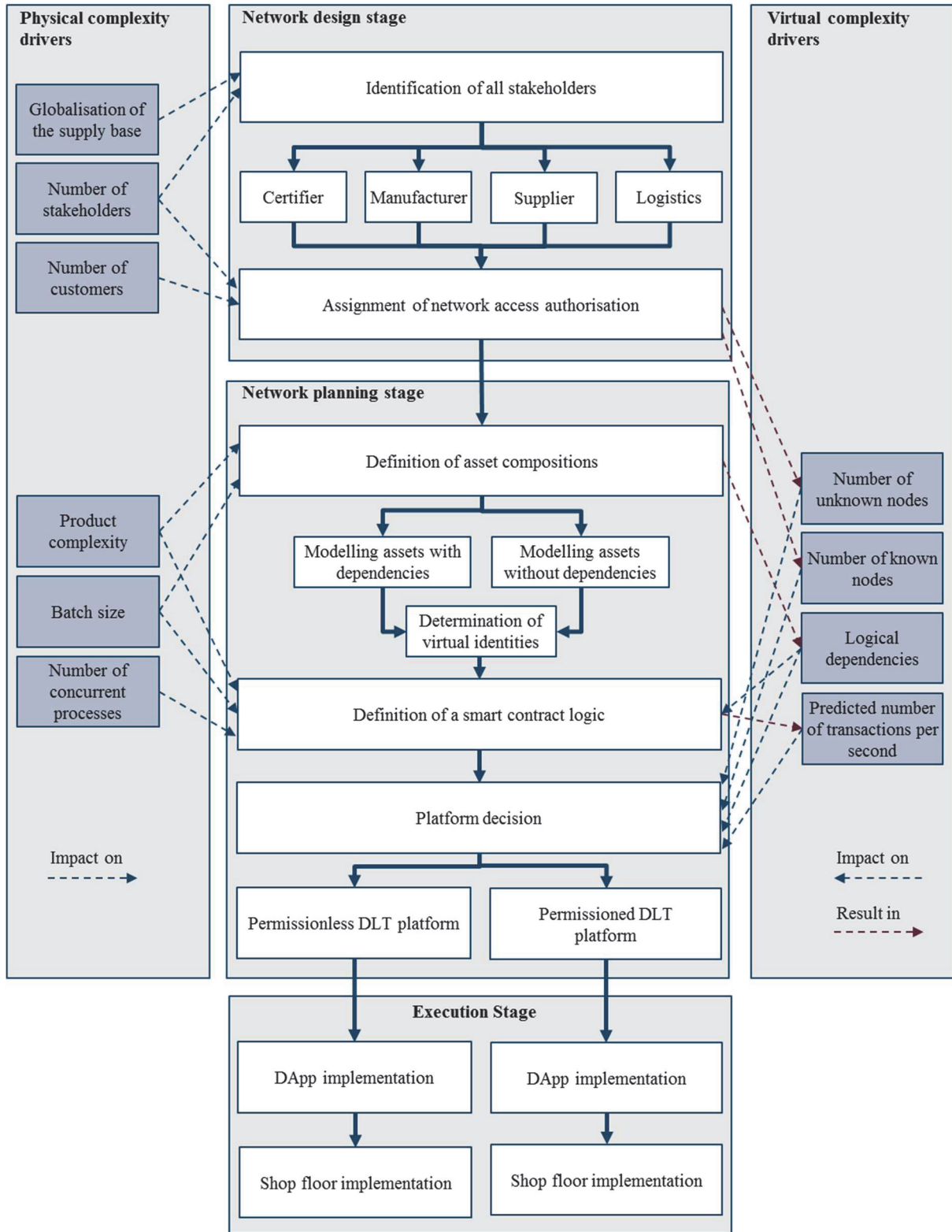


Figure 43: Conceptual framework for tracking goods in manufacturing networks using a distributed ledger technology

5 Practical implementation and dApp development

The manufacturing supply chain scenario described in Chapter 3 serves as the basis for the practical implementation. All processes of the practical implementation take place according to the developed conceptual framework. Since no similar DLT-based solution to solve the tracking and transparency problems described in Section 1.2 exists, the practical implementation is an essential part in order to validate and verify the feasibility of the framework. According to the framework, the practical implementation is divided into the network design stage, network planning stage, and execution stage.

5.1 Implementation of the network design stage

The stakeholders involved in the manufacturing process are Stellenbosch University, Werk150, ESB Business School, and an anonymous German supplier, who will be referred to as Supplier B in the course of this implementation process. Stellenbosch University is responsible of the footboard production and the German supplier for the rear fork production. Together, they represent the suppliers of the two components needed to create the assembly at the manufacturer represented by Werk150. The ESB Business School serves as a superordinate control unit, which is needed for quality assurance within the supply chain and must therefore certify each product of the suppliers and the manufacturer. In order to enable an initial implementation, the production of the footboard and the rear fork are first simulated within Werk150. The otherwise necessary intermediate delivery processes can therefore be postponed at first and the focus can be completely put on the complex virtual mapping of the physical assembly processes.

As the next step, authorisations in the network will be assigned to individual stakeholders. Here, the practical implementation distinguishes between two different scenarios. In a permissioned scenario, the customer only gets restricted access via a supernode. In a permissionless scenario, the customer gets full access to the network and is also able to operate a full node. In the permissioned scenario, the assignment of authorities is an essential element. Since the manufacturer and the certifier are essential roles within the manufacturing supply chain, these two entities operate a full node. Thus, these entities have access to the

whole distributed ledger and are able to verify transactions and network changes. In order to avoid having a central decision-maker, the network of the simplified supply chain should also consist of at least two full nodes. This guarantees a fair network stability, when the embedding of a consensus mechanism can take place. For the suppliers, on the other hand, a light node for checking the network transactions is sufficient. In the permissionless scenario, however, each stakeholder is able to operate a full node because there no participation restrictions exist. Therefore, an assignment of network authorisation can be dispensed with in this scenario. The roles of the respective stakeholders and their network authorisations in the different scenarios are listed in Table 9.

Table 9: Enlistment of all stakeholders, their roles, and their assigned network authorisations

Stakeholder	Role	Permissionless network authorisation	Permissioned network authorisation
ESB Business School	Certifier	Full node	Full node/supernode
Werk150	Manufacturer	Full node	Full node
Stellenbosch University	Supplier	Full node	Light node
Supplier B	Supplier	Full node	Light node
Customers	Customers	Full node	No node

5.2 Implementation of the network planning stage

According to the conceptual framework, the planning stage starts off with the definition of the asset's composition. The product is represented by the assembly consisting of the footboard and rear fork. For this compilation, non-financial assets in the form of certificates are necessary. A distinction is made between two different types of certificates; a certificate for the production of components and a certificate for the final assembly. Both certificates are issued by the same certifier. According to the asset categorisation described in Section 4.2.1, the certificates can only be assigned to the category of assets without dependencies, since they can be created without depending on any previous actions. The creation of the footboard, the rear fork, and the final assembly depend on successfully conducting previous actions and they are therefore categorised as assets with dependencies. The creation of the footboard and the

rear fork depend on the receipt of the respective certificates Z_1 . The manufacturing of the final assembly requires three assets, the receipt of the footboard, rear fork, and a manufacturing certificate Z_2 . These dependencies are shown in Figure 44.

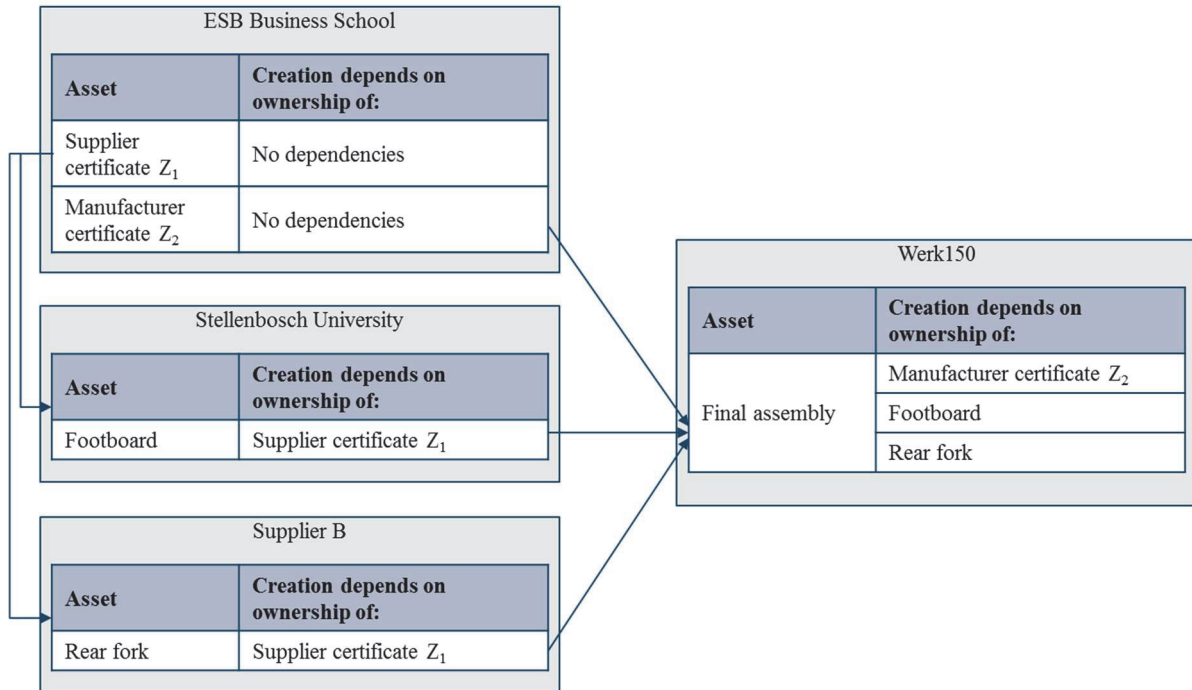


Figure 44: Dependencies of all included assets

These dependencies allow the definition of first smart contract models, which are linked to the responsible entity. The models are intended to define the logic required to create a virtual identity of assets in the form of Hash IDs on the distributed ledger. For example the ESB Business School is responsible for the creation of certificates. Therefore, the address of the ESB Business School must be linked to the smart contract model of creating the virtual identity of the certificates. After the platform decision, the real address will be created on the respective DLT platform. For the logic model itself, however, only a clear process assignment is necessary. According to the example of the certifier represented by the ESB business school, each initial asset creation of the assets shown in Figure 44 is assigned to its process owner. At the same time, these models define a first set of shared information to be stored in the Hash IDs on the network. As described in Section 4.2.1.1, these can, for example, come from already existing central systems. In this example implementation, the focus is only on the essential information of the individual assets. Thus, only the ID of the ERP system serves as the interface to the central systems. This ID is additionally supplemented with information for clear typification, such as manufacturer name and asset name. In order to guarantee the

uniqueness of each Hash ID, the timestamp of the DLT is also included during the asset creation process. A list of the assets together with all information included is shown in Table 10.

Table 10: Assets with their respective shared information

Supplier certificate Z_1	Manufacturer certificate Z_2	Footboard	Rear fork	Final assembly
Name of organisation	Name of organisation	Name of organisation	Name of organisation	Name of organisation
Name of certificate	Name of certificate	Name of part	Name of part	Name of product
Type of certificate	Type of certificate	ERP-System number	ERP-System number	ERP-System number
Creation date	Creation date	Creation date	Creation date	Creation date

The asset creation models are supplemented by a model for sending the assets. This allows the assets to be sent in a logical sequence that always alternates between creating processes and sending processes (Section 4.2.2). Thus, manufacturing processes can be realistically mapped on the distributed ledger. While the creation models are clearly assigned to the respective entity, the sending models are variably accessible to the current owner of the asset. First, the ownership of an asset is assigned to the asset's creator and is authorised to send this asset, represented by a Hash ID, to a new owner. Accordingly, only the address of the new owner can trigger further actions regarding the received Hash ID. In the example of the manufacturing supply chain, the certificates are non-physical assets, which are assigned to the suppliers and the manufacturer. Therefore, in this example the creation and sending of these assets can be combined into one model. Conversely, this also means that certificates can only be created specifically for the respective entity. Figure 45 shows the holistic smart contract model of the example supply chain in the vertically depicted chronological sequence and the alternating interrelations of creation and dispatch processes.

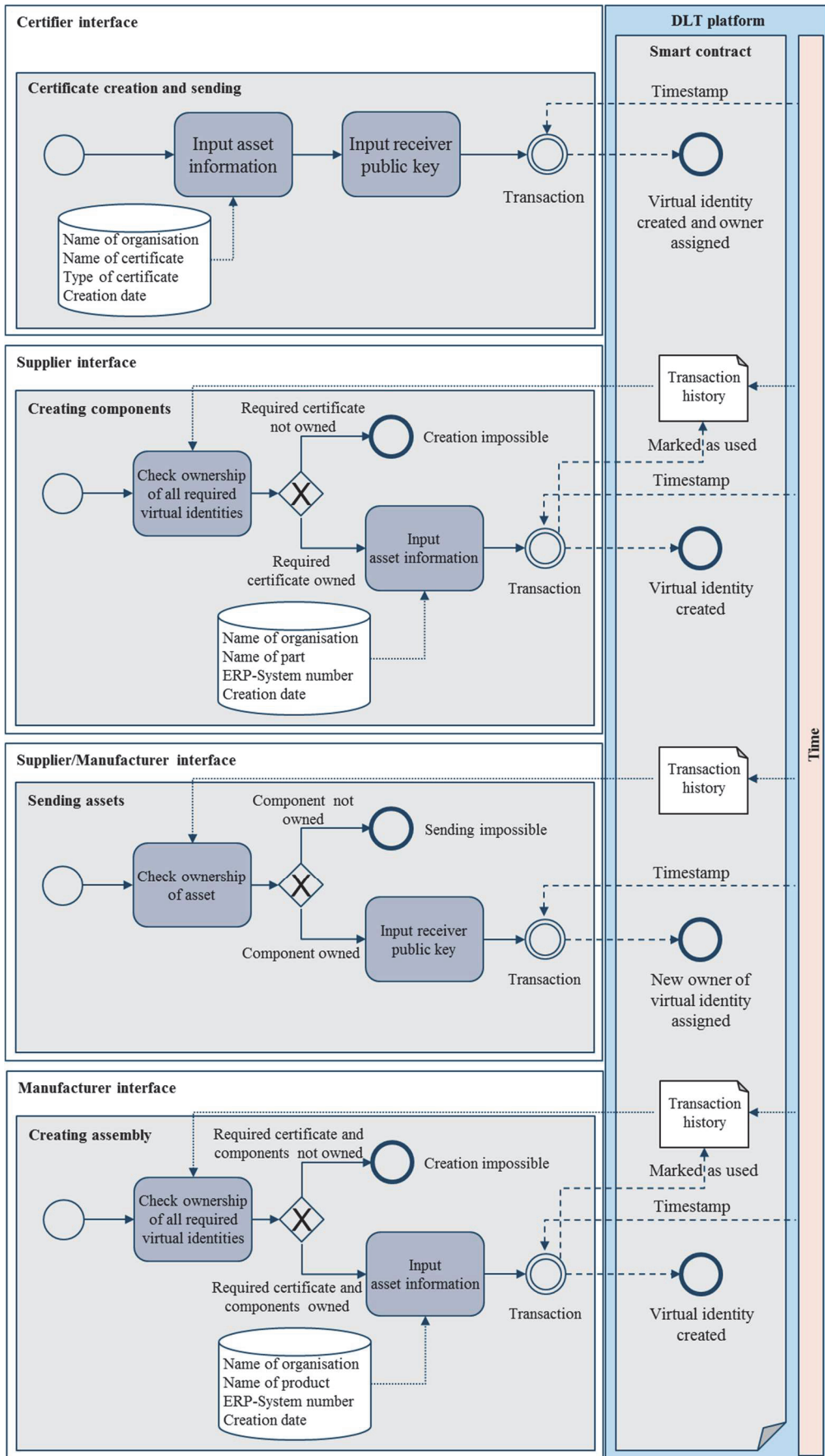


Figure 45: Holistic smart contract logic models of the manufacturing supply chain

According to Section 4.2.2.1, based on the smart contract logic, first predictions about the number of transactions per second can be made. As the logic model of Figure 45 demonstrates, every virtual asset can experience both changes of state and changes of ownership. Only the smart contract logic of creating and sending the certificates summarises the changes of state and changes of ownership in one transaction. Therefore, the holistic logical process is as follows: The certifier (ESB Business School) creates certificates and assigns their ownership directly to the suppliers (Stellenbosch University and Supplier B) and the manufacturer (Werk150). After receiving the certificates, the suppliers create the corresponding parts for the footboard and rear fork. According to the smart contract logic, the creation of these two components results in a change of state transaction on the distributed network. After a successful creation, the two components are sent to the manufacturer, which results in a change of ownership transaction. After the manufacturer Werk150 successfully receives the two components, Werk150 is able to create the final assembly, which again causes a change of state transaction. This assembly can then also be sent to a new owner. On this basis, a test scenario can be simulated with all possible concurrent processes. Based on this simulated scenario the maximum number of transactions per second of the application can be predicted. In this scenario, all functions of the smart contract logic are triggered simultaneously. This means that all entities will simultaneously create one asset and send one asset within the same time interval of one second. Therefore, the equation introduced on page 67 to calculate the prediction of transactions per seconds is used.

PTPS : Predicted transactions per second

s_n : Changes of state of asset_n

o_n : Changes of ownership of asset_n

Δt : Time interval in seconds

$$PTPS = \frac{s_1 + o_1}{\Delta t} + \frac{s_2 + o_2}{\Delta t} + \dots + \frac{s_n + o_n}{\Delta t} = \frac{\sum_{i=1}^n (s_i + o_i)}{\Delta t} \quad (1)$$

An adaption equation (1) results in the calculation below. Thereby, asset₁ refers to the certificate, where the changes of state and the changes of ownership have been summarised to one transaction. Figure 45 shows, all other entities have access to interfaces which allow the triggering of transactions caused either by a change of state or by a change of ownership. Accordingly, Asset₂ refers to all changes that affecting the component footboard. Asset₃

designates the all changes of the rear fork. Finally, $asset_4$ designates all changes of the final assembly within the time interval of one second.

$$7 \text{ PTPS} = \frac{1_1}{1s} + \frac{1_2+1_2}{1s} + \frac{1_3+1_3}{1s} + \frac{1_4+1_4}{1s} \quad (2)$$

As equation (2) shows, the triggering of all modelled smart contract functions at the same time result in 7 transactions per second. Accordingly, this value can be used to select a suitable platform in the later course of the practical framework implementation.

5.2.1 Implementation of the platform decision

As described in Section 4.2.3, a fundamental aspect of the platform decision is the Turing Completeness of the platforms to be considered. This enables the mapping of the logical links and extensive relationships of manufacturing supply chains. Therefore, platforms are first preselected according to this criterion. The BCT in particular represents the most advanced DLT in this aspect. Other DLT technologies, such as Tangle or the Block-lattice architecture, do not yet provide the embedding of smart contracts for the creation of dApps at the time of writing this work.

IOTA, a cryptocurrency based on the DLT called Tangle, targets specifically meeting the need of the Internet of Things (IoT) by adapting the DLT (Popov, 2018a). IOTA as a solution for tracking goods of global supply chains is a fundamental part of the IOTA business model (IOTA, 2019). However, at the time of writing this work, the platform IOTA does not include any tools to implement such complex applications on the platform and is only restricted to the use of IOTA as a cryptocurrency. Even though the developer team of IOTA is currently working on a solution named ‘Qubic’, a protocol that will inter alia include smart contracts, their website still states that “there are no hard dates yet” (qubic.iota.org, 2019). A similar picture can be seen at the platform Nano, a representative of block-lattice architecture technology. This platform is also limited to use as a cryptocurrency at the time of writing this work and does not provide any information about embedding smart contracts in the future (LeMahieu, 2017).

Based on the BCT, there are several platforms with embedded smart contracts supporting Turing Completeness. The largest platforms with the most comprehensive

capabilities for creating dApps are the permissionless platforms Ethereum and EOS, and the permissioned platform Hyperledger by the Linux foundation. According to the financial rating agency, Weiss Ratings, the EOS and Ethereum are currently the leading cryptocurrencies. Especially due to the scalability problems of Ethereum, EOS is more and more challenging Ethereum to become the best smart contract platform (Barclay, 2019). Linux Foundation's Hyperledger is the permissioned opponent of permissionless BCT platforms. The Hyperledger project itself, is a combination of codebase from IBM, Digital Asset Holdings, and Blockstream to create a modular design which is capable of supporting different implementations of distributed ledger-based solutions for enterprises. One aspect central to the Hyperledger design is the strong support for complex identity management with built-in certificate authority systems (Lee & Deng, 2018, p. 173). As described in Section 2.3, most of the current pilot projects combining the BCT and supply chain management are based on Hyperledger Fabrics.

To choose a suitable platform in order to practically validate and verify the feasibility of the conceptual framework, a utility analysis of the three platforms Ethereum, EOS, and Hyperledger Fabrics is conducted.

5.2.1.1 Methodology of the platform selection

The utility analysis serves as methodology to select a suitable platform. In general, utility analyses are used to make complex decisions, which require the consideration of many different aspects. Therefore utility analyses use the principle 'fragmentation', where the overall problem to be decided is broken down into subproblems. Thus, the emotional attachment to a spontaneous preference for a desired solution is dissolved and the respective partial solutions can be discussed rationally (Kuehnappel, 2019, pp. 1–4).

For the utility analysis, white papers and technical descriptions of the concerned platforms serve as the main sources. Furthermore, previous research papers are considered which have already dealt with a property comparison between the different platforms. These data are finally supplemented with different online sources to generate a recent data situation, especially regarding the scalability of the different platforms. Table 11 enlists all considered sources for the utility analysis.

Table 11: Sources used for the platform selection

No.	Sources used for the platform selection
[1]	Buterin, V. (2013). Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform.
[2]	IBM Research Editorial Staff (2018). Behind the Architecture of Hyperledger Fabric. Retrieved from https://www.ibm.com/blogs/research/2018/02/architecture-hyperledger-fabric/ (Accessed: 19 August 2019)
[3]	Blummer, T., Bohan, S., Bowman, M., Cachin, C., Gaski, N., George, N., . . . Yang, B. (2018). An introduction to Hyperledger: Creative Commons Attribution 4.0 International License. Retrieved from https://www.hyperledger.org/wpcontent/uploads/2018/07/HL_Whitepaper_IntroductiontoHyperledger.pdf (Accessed: 19 September 2019)
[4]	EOS Whitepaper (2018). EOS.IO Technical White Paper v2. Retrieved from https://github.com/EOSIO/Documentation (Accessed: 19 August 2019)
[5]	Eosio.cdt (2019). EOS Developer. Retrieved from https://eosio.github.io/eosio.cdt/latest (Accessed: 19 August 2019)
[6]	EOS Authority (2019). Block Producer Elections (Schedules). Retrieved from https://eosauthority.com/producers_schedules?network=eos (Accessed: 19 August 2019)
[7]	Wood, G. (2014). Ethereum: A Secure Decentralised Generalised Transaction Ledger - EIP-150 Revision.
[8]	Valenta, M., & Sandner, P. (2017). Comparison of Ethereum, Hyperledger Fabric and Corda. FSBC Working Paper.
[9]	Dinh, T. T. A., Wang, J., Chen, G., Liu, R., Ooi, B. C., & Tan, K.-L. (2017). BLOCKBENCH: A Framework for Analyzing Private Blockchains. ACM International Conference on Management of Data, 1085–1100.
[10]	Sajana, P., Sindhu, M., & Sethumadhavan, M. (2019). On Blockchain Applications: Hyperledger Fabric And Ethereum. International Journal of Pure and Applied Mathematics, 118(18), 2965–2970.
[11]	Njui, J. (2018). Vitalik: Ethereum (ETH) Can Scale to 500 tps Using ZCash’s ZK-SNARKs. Retrieved from https://ethereumworldnews.com/vitalik-ethereum-eth-can-scale-to-500-tps-using-zcashs-zk-snarks/ (Accessed: 19 September 2019)
[12]	Fadilpasic, S. (2019). EOS Gets New Update for Better Transaction Speed. Retrieved from https://cryptonews.com/news/eosio-gets-new-update-for-better-transaction-speed-3261.htm (Accessed: 19 August 2019)
[13]	TNW (2019). Stack Overflow Analysis: The top 10 programming languages in blockchain. Retrieved from https://thenextweb.com/hardfork/2019/05/24/javascript-programming-java-cryptocurrency/ (Accessed: 19 August 2019)

Before the decision-making process begins, a selection of the different decision alternatives is made (Kuehnepfel, 2019, p. 8). In the case the practical implementations of the framework, the platforms Ethereum, EOS, and Hyperledger Fabrics are considered. Table 12 shows a first

comparison between the basic parameters of the considered decision alternatives based on the sources shown in Table 11.

Table 12: Basic parameter comparison between the platforms Ethereum, EOS, and Hyperledger Fabrics

Characteristic	Ethereum	EOS	Hyperledger Fabrics
Governance [4,8,10]	Ethereum Developers	Block.one	Linux Foundation
Consensus [1,2,3,4,7]	PoW	Delegated PoS	Pluggable PBFT
Scalability [2,11,12]	25 TPS	4,000 TPS	3,500 TPS
Mode of operation [1,3,4,8,10]	Permissionless or permissioned	Permissionless or permissioned	permissioned
Smart Contract Language [3,5,7,10]	e.g. Solidity	e.g. C++	e.g. Go, node.js, or Java
Currency [3,4,7,10]	Ether	EOS	none

After defining the decision alternatives, the approach of the utility analysis defines relevant decision criteria. To find suitable criteria is both an analytical and a creative process, which results in a list of criteria facilitating the decision process. Once all criteria relevant for the evaluation of the alternatives have been found, the significance of each individual criterion for the final decision must be determined. The significance is expressed by means of a ratio which measures the relative significance of each individual criterion for the problem. The sum of all ratios is 100%. Although the concrete determination of the significance can be subjective, it is always comprehensible and can be justified at any time (Kuehnappel, 2019, pp. 8–18). For the selection of a suitable platform, the criteria can be derived from the conceptual framework. In this context, it is important to mention that the selection of a platform aims to practically validate and verify all aspects of the conceptual framework. The chosen platform should ensure this as holistically as possible, resulting in a more comprehensive fulfilment of criteria. The list of the criteria and their weightings are explained below. The number in brackets represents the respective ratio in percent. Since all three platforms are open source, the price does not have to be considered as a decision criterion.

- 1.) *Suitability of programming languages (30%)*. According to the platform decision flow (p. 75), the Turing Completeness represents an essential aspect when implementing complex smart contract-based applications. The programming languages accepted by the respective platform must therefore also be able to map non-financial applications. Since the implementation of the framework is a completely new application without any existing comparable programs, the knowledge of the programming language and its functions should be as easily accessible. Since the success of the implementation depends significantly on this criterion, the ratio is set at 30%.
- 2.) *Scalability (30%)*. The platform decision flow shows that the scalability of the platform has a major impact on the platform's chosen operation mode. At the same time, insufficient scalability leads to complications in the temporal sequence of the triggered transactions. To ensure a smooth implementation, the platform should be able to process at least the predicted number of transactions per second. Since a too low scalability can lead to significant disruptions in the process, the ratio is also set to 30%.
- 3.) *Decentralisation/Security (20%)*. As described in Section 4.2.3.1, the decentralisation of a platform is directly related to the architecture of its consensus mechanism. When only a low number of nodes is involved in the transaction validation process, these nodes provide critical vulnerabilities to the network's security. Although such weaknesses have no direct influence to prove the practical feasibility of the framework itself, it questions the security in terms of the repeatedly mentioned immutability when tracking goods by adopting a BCT. Therefore, this criterion is weighted at 20%.
- 4.) *Use as a permissionless platform (10%)*. The framework proposes two possible different operation modes. Therefore, the practical implementation must either be tested and validated based on the implementation of a permissionless platform, or on the implementation of a permissioned platform. In the best case scenario, one platform can be operated in both modes. If the advantages of the previous criteria predominate to such an extent even though a platform can only be operated in one of the two operation modes, an implementation of two different platforms can be considered; one platform for permissionless implementation and one for permissioned implementation.

Since such a decision depends strongly on the score of the previous criteria, this criterion receives a ratio of 10%.

5.) *Use as a permissioned platform (10%)*. The considerations of the previous criterion also apply to the use of a platform in a permissioned operation mode. If the advantages of the first three criteria predominate to such an extent even though a platform can only be operated in a permissioned operation mode, an implementation of a permissioned platform can be considered to specifically verify and validate this aspect of the framework. Therefore, the ratio of this criterion is also set to 10%.

In the utility analysis, the individual criteria are evaluated according to a defined evaluation scale (Kuehnappel, 2019, pp. 16–20; Schlink, 2019, pp. 428–441). For the utility analysis to select a suitable platform, a five-point rating scale was chosen, with 0 representing the worst rating and 5 representing the best rating. The exact rating scale and the specific meanings of each score are listed in Table 13.

Table 13: Rating scale used for utility analysis

Score	Meaning
0	Criterion not met
1	Criterion is inadequately met with significant deficiencies
2	Criterion is sufficiently met with deficiencies
3	Criterion is met to a good extend with some limitations
4	Criterion is largely met with slight limitations
5	Criterion is fully met

As the final step of the utility analysis, each score for each criterion is multiplied by the ratio of the respective criterion. The sum of all evaluated criteria results in the final score of the respective decision alternative. The decision alternative with the highest score represents the solution that meets the requirements to the greatest extent (Kuehnappel, 2019, pp. 16–20; Schlink, 2019, pp. 428–441). The final result of the utility analysis is presented in Table 14.

Table 14: Result of the utility analysis in order to select a suitable platform

Criterion	Ratio	Ethereum		EOS		Hyperledger Fabrics	
		Rating	Score	Rating	Score	Rating	Score
Suitability of programming languages	30%	5	1.5	3	0.9	4	1.2
Scalability	30%	4	1.2	5	1.5	5	1.5
Decentralisation/Security	20%	5	1	4	0.8	3	0.6
Use as a permissionless network	10%	5	0.5	5	0.5	0	0
Use as a permissioned network	10%	5	0.5	5	0.5	5	0.5
Result	100%	4.6/5		4.1/5		3.8/5	

As the result shown in Table 14 shows, the Ethereum platform with a total score of 4.6 is the preferred solution in order to practically implement the conceptual framework. In the first criterion, the suitability of the programming language, Ethereum is the only platform that receives a score of 5 points. Ethereum is the only platform in this comparison, that supports with ‘Solidity’ a programming language specifically designed to develop smart contracts [1,7]. Additionally, this programming language is also the most popular programming language for blockchain developers according to an analysis of the ‘StackOverflow’ software development platform [13]. Hyperledger Fabrics supports several programming languages, whereby JavaScript is the second most popular programming language among software developers [3,13]. As a result, Hyperledger Fabrics receives a score of 4 points in this category. EOS only allows users to program smart contracts in C++, which is considered to be a cumbersome programming language by blockchain developers [5,13]. Therefore EOS gets only 3 points in this category.

In terms of scalability, all platforms are capable of processing the predicted transactions per second of the model supply chain. However, the Ethereum platform with its 25 TPS, is already quite close to calculated 7 TPS. Even the supply chain for the application has deliberately been simplified. With 4,000 TPS (EOS) and 3,500 TPS (Hyperledger Fabrics), the other two platforms are well above the requirements [2,11,12]. Therefore, only Ethereum with a score of 4 points, does not get the maximum score in the Scalability criterion.

The PoW consensus algorithm of Ethereum represents the most decentralised consensus mechanism among the compared platforms [1]. A ‘51% attack’ (described in Section 2.2.2.3), at the Ethereum network actually means a simultaneous manipulation of 51% of *all* nodes participating at network. As a conclusion, this mechanism can be considered as extremely secure. EOS uses the delegated PoS algorithm, which also represents a decentralised consensus algorithm. Unlike Ethereum, this does not involve the entire network, but only a selected group of 21 nodes in the block creation process [6]. Consequently, Ethereum receives a score of 5 points and EOS a score of 4 points in this criterion. In Hyperledger Fabrics, particularly PBFT-based consensus mechanisms are involved. PBFT mechanisms assume that all identities of the participating nodes are known, which leads to some reservations, especially in terms of decentralisation [9]. Therefore Hyperledger Fabrics receives 3 points in this criterion.

The evaluation of the use as a permissionless network is relatively straightforward. Since Ethereum and EOS are both Platforms that were originally designed to be open to the public, these two platforms are perfectly suitable for the use as permissionless networks [1,4]. Hyperledger Fabrics however, can only work reliably in a permissioned set up, due to its PBFT-based consensus mechanism [9]. Accordingly, this criterion cannot be met by Hyperledger Fabrics. Conversely, Hyperledger Fabrics is ideally suited for use as a permissioned platform. This also applies to the EOS and Ethereum platforms, since public networks are also suitable for use as permissioned networks [8,10]. Additionally, as described in Section 4.2.3.1, the consensus algorithm PoW can be executed efficiently by standardising the ‘difficulty’ value of the algorithm in permissioned networks.

In summary, the Ethereum platform with its own programming language, Solidity, offers an excellent possibility for the practical implementation of the framework. The platform compensates for the low scalability through a high degree of decentralisation and security. In addition, the platform is suitable for use as a permissionless as well as a permissioned platform. Consequently, in this research the practical verification and validation of the proposed framework will be based on the Ethereum platform. The public Rinkeby network serves as a permissionless environment. In order to have a permissioned environment available, an own permissioned Ethereum network is set up.

5.2.2 Set up of a permissioned Ethereum network

Based on the outcome of the network design stage and network planning stage a private Ethereum network will be set up. First of all, the appropriate hardware is selected in order to guarantee a network structure as described on page 86. As described during the design stage, the manufacturer and the certifier are essential roles within the manufacturing supply chain and are therefore represented on the network by operating full nodes. Due to the higher processor performance of full nodes, the full nodes in the permissioned network consist of desktop computers with an Intel Core i7 (2.90 GHz) processor and 16 GB RAM. The manufacturer's node (Werk150) is used as a supernode to give customers access to the blockchain data. For the suppliers a light node for checking the network transactions is sufficient. The light nodes of the network each consist of Raspberry Pi 3 with a processing speed of 1.40 GHz and 1 GB RAM. On all systems the Ethereum client called 'Geth' is installed.

```

1. {
2.   "nonce": "0x00000000000000042",
3.   "mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",
4.   "difficulty": "0x400",
5.   "alloc": {},
6.   "coinbase": "0x0000000000000000000000000000000000000000000000000000000000000000",
7.   "timestamp": "0x00",
8.   "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
9.   "extraData": "0x436861696e536b696c6c732047656e6573697320426c6f636b",
10.  "gasLimit": "0xffffffff",
11.  "config": {
12.    "chainId": 42,
13.    "homesteadBlock": 0,
14.    "eip155Block": 0,
15.    "eip158Block": 0
16.  }
17. }
```

Listing 1: Source code of the genesis file

To generate the first block of blockchain, a 'genesis.json' file is created. This file defines all important characteristics of the network. The analysis conducted in Section 4.2.3.1.1, concludes that the determination of a fixed 'difficulty' can reduce the required processor speed in permissioned PoW-based blockchain networks. Listing 1 shows the source code of the genesis file. As line 4 states, the difficulty of the network is set to a value of "0x400". This low value enables the 2.90 GHz processors a fast transaction validation with low energy consumption at the same time. In line 12, the genesis file defines the ID, which allows other

nodes to connect to the network. The ID “1” for example, is already used by the Ethereum main network. The permissioned network for this work uses the network ID “42”.

The difference from a permissionless network is, that other nodes can only join the network, when their address is added to a ‘static-nodes.json’ file. This file is shared among all network participants with a list of all participating nodes. Only if all nodes have the same ‘static-nodes.json’ file stored locally in their Ethereum blockchain folder, will they be able to connect to the platform. This ensures that only selected nodes are able to join the network. Each node of the network is able to create accounts consisting of a pair of public and private keys. A list of all entities, their roles, and the corresponding Ethereum address is shown in Table 15.

Table 15: Roles and their corresponding Ethereum addresses

Stakeholder	Role	Public key
ESB Business School	Certifier	0x9ef7a9150c314104125F2265904C5d1A91560562
Werk150	Manufacturer	0x61940bdc9daf18b0F83F9d30F527416f375F1A6C
Stellenbosch University	Supplier	0x7DFFbFf611D60B941545AfeD3a6F33bfa1F58521
Supplier B	Supplier	0x9036Ba110D503D1270d92f06e58bB28d3b1f3Be1

5.3 Implementation of the network execution stage

This section describes the programming and implementation of the dApp, as well as the implementation in the shop floor. The source codes are attached to the Appendix of this work. This section only focuses on the most essential elements of the source code and their functions. Finally, the description of the operational process is presented.

5.3.1 Development of the dApp

The smart contract forms the basis of the dApp and is programmed in Solidity programming language. The defined roles must be connected to their addresses when deploying the smart contract on the network. This enables all entities to later have access to their functions such as

creating assets. As Listing 2 shows, the roles Certifier, Supplier A, Supplier B and the Manufacturer are defined in this smart contract. To enable that, the manufacturer is, for example, represented by Werk150 in the application, their address shown in Table 15 must be linked to this role ‘manufacturer’ at the moment of the smart contract deployment. This is an essential part of the smart contract, since it defines that an entity only has access to functions intended for specifically that entity. To manage the public and the private key of each entity, this approach uses the open source ‘Metamask’ Ethereum wallet.

```

1. pragma solidity >=0.5.8;
2. contract SupplyChain{
3.     address certifier;
4.     address supplierA;
5.     address supplierB;
6.     address manufacturer;

```

Listing 2: Source code for defining the roles at the moment of smart contract deployment

In order to be able to trace logical dependencies within the smart contract, Solidity offers the so-called ‘mapping’ reference type. Mappings can be seen as hash tables which are virtually initialised. Therefore, the key data is not stored in a mapping, only its hash to look up the value. Since Ethereum uses the Keccak-256 hashing algorithm, this hash is formatted according to the algorithm’s specific format. This makes the ‘mapping’ reference type suitable for managing the Hash IDs and their history. In addition, a memory is created for each role, which is used to store the Hash IDs owned by the respective role. With a simple ‘get’ query function, any entity can have access to the status of the owned Hash IDs at any time. This connection is shown in the following source code. In this context, Listing 3 shows as an example the memory of all certificates owned by Supplier A and the corresponding ‘get’ function.

```

1. bytes32[] govtToSupplierACertificates;
2.
3. function getSupplierACert() public view returns (bytes32[] memory){
4.     return govtToSupplierACertificates;

```

Listing 3: Connection between memory and ‘get’ function in the source code

Based on this architecture, the product compositions can be transferred to the smart contract. Listing 4 shows the source code of the final assembly’s product composition, which was defined in the smart contract logic model (Section 5.2). Lines 7-9 define that the product

consists of a Component A from Supplier A, a Component B from Supplier B, and a Certificate from the Certifier.

```

1. struct Product{
2.     string nameOfProducer;
3.     string typeOfProduct;
4.     string serialId;
5.     string date;
6.     bytes32 certificateHash_p;
7.     bytes32 certificateHash_sup_a;
8.     bytes32 certificateHash_sup_b;
9.     bytes32 certificateHash_govt;
10. }

```

Listing 4: Source code of the assembly's products composition

The Hash IDs are also created by the Keccak-256 hashing algorithm. To guarantee the uniqueness of each Hash ID, the blockchain's timestamp as a variable is included when hashing all information. This is shown in Listing 5 at the example of Component A, representing the footboard. The source code was simplified to illustrate the essential aspects of the procedure. The product information has been summarised and is therefore written in italics. The whole source code can be found in Appendix B. Line 2 shows the involvement of the timestamp function when hashing the information with the Keccak-256 algorithm. Line 3 specifies that only the address of Supplier A has access to the function. In Line 4, the status of the involved certificate is set to 'used'. This guarantees that the same certificate cannot be used twice. Finally, in Line 5 the new Hash ID is added to the memory of the parts owned by Supplier A.

```

1. function CreateComponetA(Product Information)
2.     public{bytes32 certHash = keccak256(abi.encodePacked(Product Information,block.timestamp));
3.     if(_address == supplierA){
4.         govtToSupplierAHistory[_certHash].used = true;
5.         supplierAParts.push(certHash);
6.     }
7. }

```

Listing 5: Simplified source code for the creation of the Hash ID of Component A

The history of the assembly's Hash ID to enable tracking and auditability of all involved assets can be checked with a function summarising the ownership history of each component (Listing 6).

```

1. function getCertificateHistory(bytes32 _certHash) public view returns (bytes32,bytes32,bytes32,bytes32){
2.     return(productDetails[_certHash].certificateHash_p,
3.         productDetails[_certHash].certificateHash_sup_a,
4.         productDetails[_certHash].certificateHash_sup_b,
5.         productDetails[_certHash].certificateHash_govt);

```

Listing 6: Source code to receive the history of the assembly

In order for users to have access to the smart contract and its functions, a GUI must be set up. In this approach, the interface ‘ReactJS’, a JavaScript library for building user interfaces was selected. To connect the ReactJS GUI with the smart contract and to enable an interaction, the Ethereum platform requires a specific Contract Application Binary Interface (ABI) stored in JSON format. Therefore, the encoding is carried out according to the schema of the Ethereum Revision (2019). According to this scheme, for example the Solidity source code shown in Listing 6, results in the JSON source code ABI of Listing 7.

```

1.  "name": "productDetails",
2.    "outputs": [
3.      {
4.        "name": "nameOfProducer",
5.        "type": "string"
6.      },{
7.        "name": "typeOfProduct",
8.        "type": "string"
9.      },{
10.       "name": "serialId",
11.       "type": "string"
12.     },{
13.       "name": "date",
14.       "type": "string"
15.     },{
16.       "name": "certificateHash_p",
17.       "type": "bytes32"
18.     },{
19.       "name": "certificateHash_sup_a",
20.       "type": "bytes32"
21.     },{
22.       "name": "certificateHash_sup_b",
23.       "type": "bytes32"
24.     },{
25.       "name": "certificateHash_govt",
26.       "type": "bytes32"

```

Listing 7: JSON ABI resulting from the history function of the assembly

The source codes of the smart contract and the respective ABI can be found in the Appendix of this work.

5.3.2 Operational process description

The operative process starts with the preparation of certificates. Since the certificates represent non-physical assets, the creation takes place only virtually. Figure 46 illustrates the sequence of all necessary steps in order to create valid certificates on the Ethereum blockchain. As the first step (1), the Certifier represented by the ESB Business School, fills in all information required to specify the certificate. Next (2), the Certifier chooses the type of certificate. Either it is a certificate for the suppliers, or a certificate for the manufacturer. As described in Section 5.2, the certificate creation and the certificate sending process, are put together into one transaction. Therefore, the address of the receiver must be selected as the third step (3). Finally (4) the certificate can be created and sent to the selected receiver. Pressing this button results in a transaction on the blockchain network. The certificate creation is performed until each entity is provided with at least one certificate.

Figure 46: Process sequence of the certificate creation

After the suppliers have received their certificates, they can start the production of their components. The process is illustrated in Figure 47 using the example of the footboard producer represented by Stellenbosch University. The procedure for the supplier of the rear

fork is very similar, since both suppliers use the same interface. First the physical component, the footboard, is produced. After the production process has been successfully completed and the quality has been checked, the virtual identity of the footboard can be created on the blockchain. For this purpose, the information specifying the footboard is filled in on the interface (5). The entered ID corresponds to the footboard's ID from the ERP system. In order to be able to create the Hash ID of the footboard, first the previously received certificate must be selected (6). Afterwards, the creation of the Hash ID can be initiated (7). After its creation, the Hash ID is located in the 'virtual inventory' of the supplier, which is connected to the wallet of the supplier. By attaching this Hash ID to the physical component, the component can be linked to its virtual identity. For the attachment of the Hash IDs to the components and the assembly, the QR Code technology was chosen. On the one hand, both the footboard and the rear fork have large areas with enough space to attach readable QR codes and on the other hand, QR code printers can easily be integrated into the production process. Furthermore, the camera of almost every smartphone can be used as a QR code reading device. The QR code stores only the Hash ID, which enables clear identification of each component (8). The component can now be sent to the manufacturer simultaneously in both 'worlds', physical and virtual, on the blockchain. In order to send the Hash ID virtually, the address of the manufacturer must be typed in as receiving address (9), which corresponds to the address of Werk150 in Table 15 on page 100.

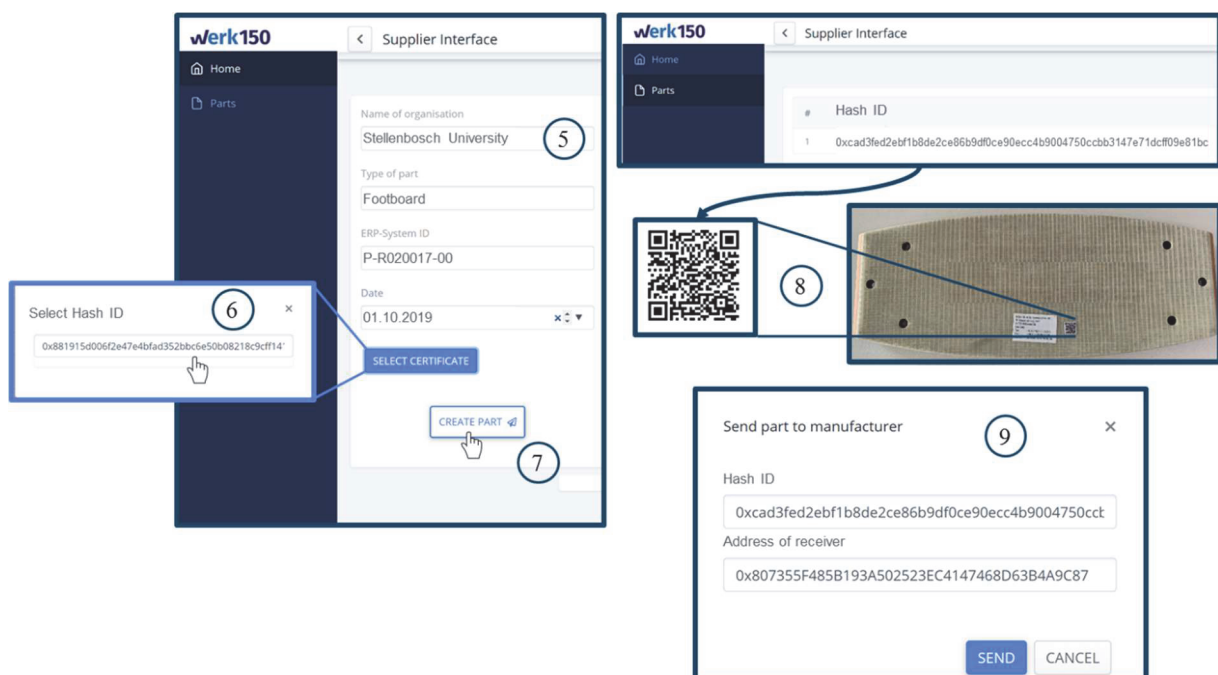


Figure 47: Process sequence of the component production

The manufacturer can start to assemble the two components if a certificate, the footboard, and the rear fork are in his physical inventory as well as in his virtual inventory. This process sequence is illustrated in Figure 48. Before each step, it is essential to clearly identify each component to avoid mistakes and to ensure a smooth virtual mapping of the process. Therefore, the Hash ID of each component must first be verified with a reading device (10). After the identification, the two parts can be assembled physically and virtually. As with all previous assets, the assembly's specifying information must first be filled in (11). Additionally, the Hash IDs of all used components must be selected (12). Since this process is not reversible, the used components should be checked carefully, to guarantee a matching of the physical and virtual process. Lastly, the newly created Hash ID must be converted to a QR code and attached to the assembly (13).

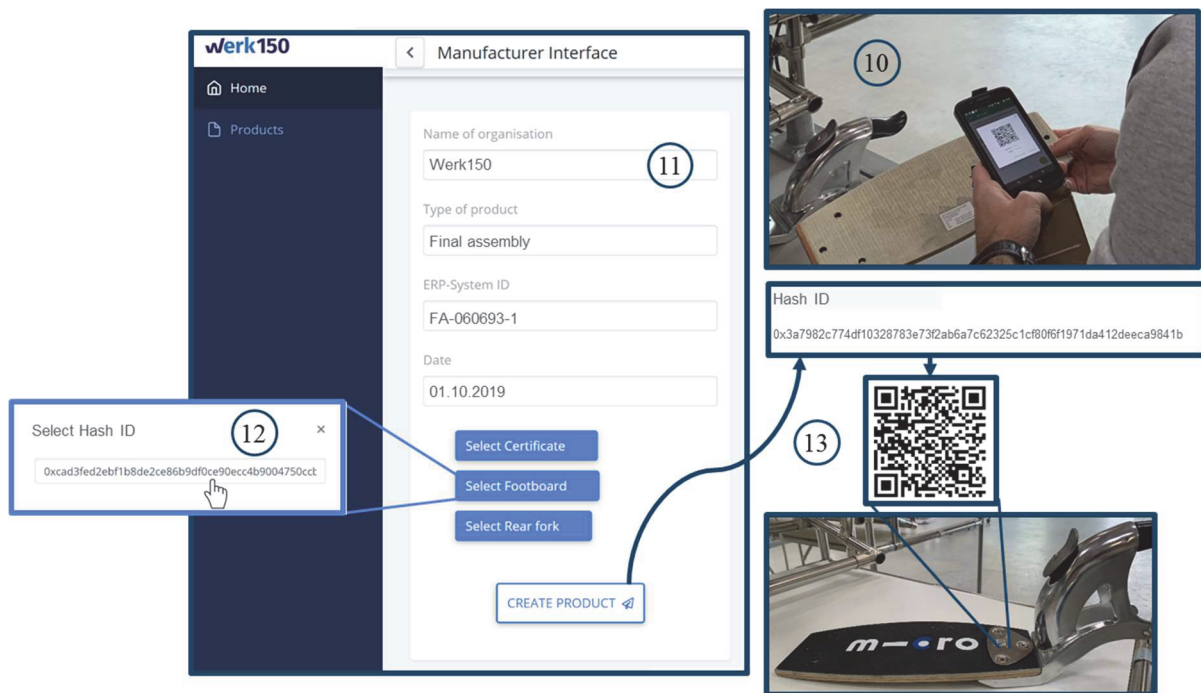


Figure 48: Process sequence of the assembly manufacturing

All information about every assembly can be retrieved via a user interface. As Figure 49 shows, the interface provides the user with information about the ownership, producer identity, and composition of the assembly. Although all values are given in the Ethereum typical hexadecimal format, it illustrates the correspondence of the data. For example the address of the producer corresponds exactly to the address of Werk150 shown in Table 15 on page 100. Also, the Hash ID of the involved footboard matches the Hash ID of the footboard created in Figure 47. The components are thus fused in a 'virtual assembling' process to form a new

identity, which represents the identity of a new product. At the same time, this new identity contains the information of all involved assets and entities, which enables not only a tracking but also the immutable auditability of every single component on the blockchain.

The screenshot shows a user interface window with a close button (X) in the top right corner. The window displays the following information:

- Currently owned by: 0x61940bdc9daf18b0F83F9d30F527416f375F1A6C
- Address of producer: 0x61940bdc9daf18b0F83F9d30F527416f375F1A6C
- Assembly – Hash ID: 0x3a7982c774df10328783e73f2ab6a7c62325c1cf80f6f
- Footboard – Hash ID: 0xcad3fed2ebf1b8de2ce86b9df0ce90ecc4b9004750cct
- Rear fork – Hash ID: 0xd647d48eacacb4a7eb68e621e5457bb0cfbf267fb35d
- Certificate – Hash ID: 0x18bf7f13b213b70467878d7f9828b507d1e8e958efa5

Figure 49: User interface to retrieve all information about the composition and the history of an assembly

The access to the interface shown in Figure 49, represents the main difference between running the dApp on a permissionless or permissioned network. On a permissionless blockchain, any address can access this interface and track the composition of the individual assets. In a permissioned environment however, only the entities with permission can access the interface. Nevertheless, the information can be processed for all entities that do not participate at the network and made accessible, for example, via another application. As described in Section 4.1.2, this function can be taken over by a supernode. However, it must be noted that this data can no longer have the immutability characteristics of DLTs. Accordingly, the data can be manipulated by the supernode. Therefore, the transparency guaranteed with such a construct can only be regarded as restricted transparency for a customer or other entities that are not participating in the network.

6 Verification and validation of the framework

The framework combines a vertical conceptual SCM system implementation process with horizontal influencing complexity drivers. As a result, the framework proposes an adoption of the DLT in order to enable the tracking of manufactured goods and all components they consist of. Since the framework proposes an entirely new approach by adopting the DLT, its feasibility is verified and validated based on the results of the practical implementation. A central element of this framework is represented by the implementation of smart contracts. These serve as a foundation for creating dApps, which are able to map complex relationships and logical dependencies of manufacturing supply chains. Therefore, the immutability and security of the smart contract will first be verified and validated. Subsequently, it is verified and validated to what extent the practical blockchain-based implementation enables the tracking of manufactured goods. Finally, the achieved customer transparency is verified and validated.

6.1 Smart contract immutability and security

From the moment of smart contract deployment, the source code of the whole dApp is unchangeable. In this context, no differences between permissioned and permissionless networks were to be found. On both networks, smart contracts receive a unique address when deploying them on the network, which allows the interaction with the smart contract. Therefore, all users accessing to the same smart contract address always use exactly the same source code. Retroactive changes or manipulations can be considered impossible, because the deployment of a modified source code inevitably results in a completely new smart contract on the network, represented by an entirely different and new address. This makes a secret changing or updating of the source code impossible.

Due to the immutability of the source code, security mechanisms must be defined within the source code itself. Therefore it is necessary to firmly link the address of each respective entity to the functions specifically intended for a certain entity. Since only the respective entity has the appropriate private key, this entity is the one capable of triggering actions linked to its address. This mechanism could not be circumvented in the practical implementation of the framework. Any attempt to access the functions intended for a certain

entity with the address of a wrong entity is immediately interrupted by an error message. The error resulting from such an attempt is shown in Figure 50.

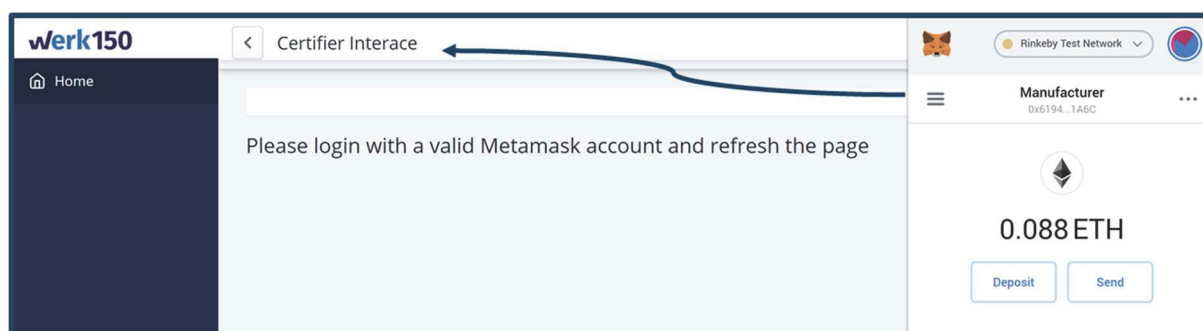


Figure 50: Attempting to access a functions with an unauthorised address results in an indispensable error of the system

Another protective mechanism is the precise specification of dependencies within the source code of the smart contract. The different certificate types of the certifier provide a suitable example of this. The certifier is on the one hand entitled to create certificates for the suppliers and on the other hand to create certificates for the manufacturer. The source code excludes the possibility of the certifier providing a supplier with manufacturing certificates. Accordingly, such an attempt will result in an error message and the prevention of such transaction. Thus, by defining clear dependencies within the source code, inconsistencies can be prevented and excluded in advance. At the same time this also shows the enormous importance of the accuracy during the network design stage and the network planning stage of the framework. Due to the immutability of smart contracts, unnoticed dependencies can lead to significant security vulnerabilities which cannot be easily removed in the stage of execution.

The practical implementation of the framework proves that the immutability can be seen as a given component of DLT-based smart contracts with a significant benefit in the security of such applications. In order to guarantee a secure operation, the structure of the smart contract's source code itself is of importance. In this code, all processes must be clearly linked with the responsible entities to create clear ownerships of the smart contract functions. Additionally, clearly defined dependencies help to avoid errors and discrepancies within the application. This requires a detailed network design and network planning because retroactive changes to the smart contract are complex due to its immutability after having been deployed on the network. In order to minimise this problem, it is possible to develop mechanisms within the source code, which allow retroactive changes without questioning the decentrality of the

application. An example for this could be the use of complex decentralised voting mechanisms embedded into the source code of the smart contract, which would give every participant the opportunity to vote on possible changes or updates.

6.2 Tracking manufactured goods by using a distributed ledger technology

To enable the tracking of manufactured goods by using a distributed ledger technology the framework proposes the establishment of a virtual identity representing assets on the distributed ledger. In doing so, the transactions resulting from an asset change of ownership and change of status can be recorded on the distributed ledger to verify its position and status. As a virtual identity the framework elaborates the approach of creating Hash IDs, which logically result from hashing input data in order to provide initial information about the origin, the composition, and the time of creation of the asset they refer to. The respective hashing algorithm of the selected platform is used for creating the Hash IDs. Including the timestamp when creating a Hash ID guarantees its uniqueness which results in each asset having a unique virtual identity.

As the practical implementation is demonstrating, this approach enables the successful creation of unique virtual identities. Since the Ethereum platform was chosen for the implementation, the hashing algorithm Keccak-256 is used to create the Hash IDs, consisting of 64 hexadecimal characters. The uniqueness is guaranteed by the hashing algorithm itself. This allows the theoretical creation of 2^{128} different virtual identities (Bertoni et al., 2013). Accordingly, the chance of two identical Hash IDs can be considered as practically impossible. In the practical implementation, the barcode technology in form of QR codes was chosen to link the Hash ID with the physical asset. This solution effectively implements the linking of physical components to their virtual identities. Especially due to the fast creation of QR codes and the availability of reading devices, this technology can easily be integrated into production operations. The practical implementation shows that the identification of each component before every assembly step is important. During the validation process, the identification of components was carried out by the employees. However, this also slows down operational processes. The use of robots could speed up this process. Robots could identify components and pre-sort them before handing them over to the employee.

Similar to financial transactions, the Hash IDs can be sent in order to change their authorised owners on the distributed ledger. Each change of ownership is carried out in the form of a transaction on the network. The information about the virtual owner of the Hash ID thus provides information about the current location of the asset. In addition, Hash IDs can be merged virtually, for example, when components are assembled. This status change also results in a transaction on the network. Thus not only the location and the owner of an asset, but also the status and composition can be identified. As the practical implementation of the framework proves, a DLT-based supply chain application goes beyond the traditional use of tracking solutions. Since all transactions can be traced at any time, the entire history of an assembly is verifiable for all network participants. The result is a database-like structure that logically links the virtual identities in the course of time. Every new identity is a logical result of all virtual identities it consists of. This allows tracking of the history of all manufactured goods and all components they consist of. This relationship is shown in Figure 51.

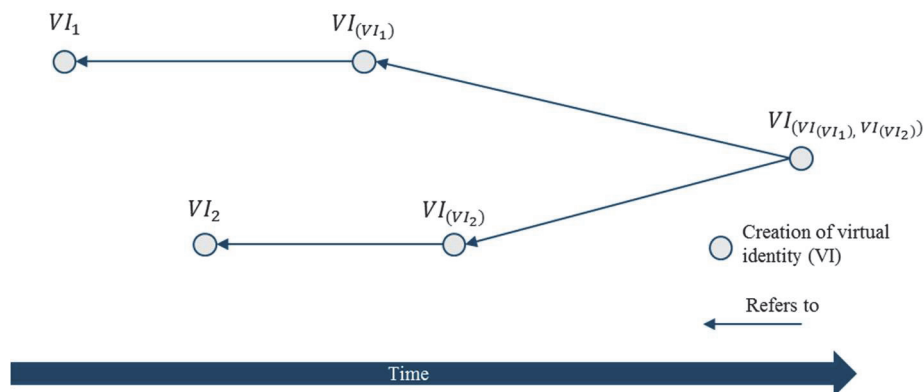


Figure 51: Database structure linking the virtual identities in a course of time

Not only can every component be traced virtually but also it also can be checked for consistency with the physical components. In practice this interaction between the virtual identity and their physical counterpart proves to be effective and precise. During the validation process, 100 assets were created on a permissioned and a permissionless network. All assets and their composition could be traced back error-free via the interface. However, there were still differences between the permissioned and permissionless applications in terms of temporal accuracy.

6.2.1 Time accuracy differences between a permissioned and permissionless application

To verify the accuracy of DLT-based tracking processes, 100 assets are created on both the permissionless and permissioned network. As described before, the creation of virtual assets results in transactions on the distributed network. For the verification, the time between the triggering of a transaction and its confirmation is measured. This is particularly interesting as the blockchain is often connected with the possibility of enabling real-time tracking within supply chains (Frost & Sullivan, 2019, p. 12).

As a test environment for the permissionless network the application is put on the Ethereum Rinkeby test network (Section 5.2.1). As test environment for the permissioned network, their own Ethereum network described in Section 5.2.2 is used. On the Rinkeby test network free test Ether can be used. The 100 transactions on the Ethereum main network, however, would have caused transaction fees of about \$0.4. On the own permissioned network, there are no transaction fees for all network participants.

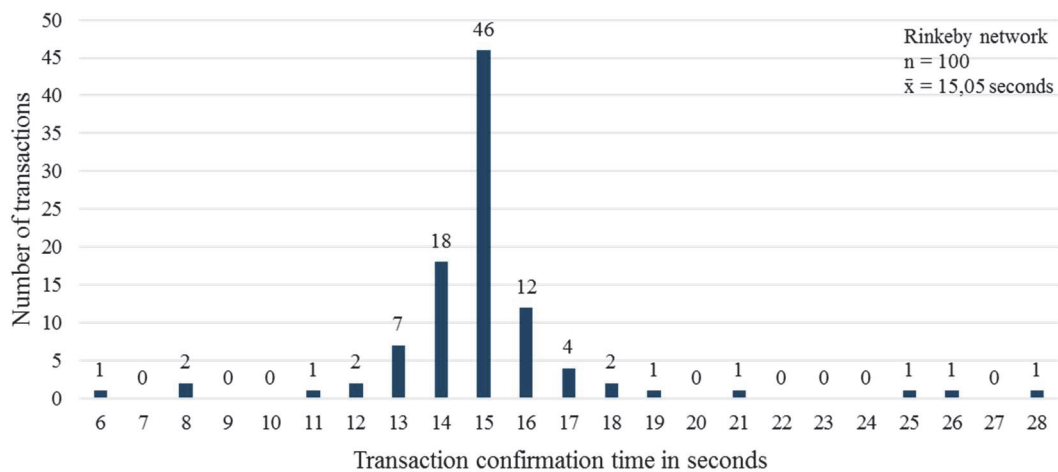


Figure 52: Transaction confirmation times of the Ethereum Rinkeby network

Figure 52 shows the distribution of the transaction confirmation times in a permissionless environment. As the figure shows, the state of an asset and its location on the Rinkeby network, can be tracked with a mean accuracy of 15.05 seconds. The shortest confirmation took 6 seconds and the longest confirmation 28 seconds. In the case of the practical implementation these values do not lead to bottlenecks, as there is significantly more time between the

individual processes. But as stated in Section 6.2.1, the framework also considers the possibility of a more detailed tracking for example with each machine within a production line having its own address on the network. If a component changes its position between two machines in the physical world, the component also passes into virtual possession from one machine to the other. Thus the exact location of an asset can be tracked within the production line by tracking the virtual possessions. However, as the test results show, this can only be done on a permissionless network if there is enough time between each process for the network to confirm the transactions. For example, it is not possible for an asset to change the owner if the creation of this asset has not yet been confirmed by the network. Since the virtual and physical processes must run simultaneously, such a delay would cause disruption of the entire production process. Therefore the time intervals between the processes cannot be based on the mean transaction confirmation time. The time intervals between the processes must at least be as high as the maximum confirmation time in order to avoid conflicts. In case of the tested example manufacturing supply chain, the time intervals between each process must therefore be at least 28 seconds on the permissionless Rinkeby network. This is the only reliable way to avoid procedure bottlenecks caused by delayed transaction confirmations.

In the permissioned network, the entire capacity of the network is available to the decentralised tracking application. At the same time, a low and fixed level of difficulty is defined in the genesis block of the blockchain. This allows the mining nodes to confirm transactions within a second or less, which allows a very accurate tracking of the status and location of each asset. Accordingly, the scalability of the permissioned Ethereum platform does not limit the usage of the application.

The result illustrates how strongly the scalability of the DLT - in this case especially of the blockchain technology – limits the realisation of real time tracking. Applications based on the proposed framework can currently only enable real-time tracking reliably in permissioned networks. Therefore, the time intervals between processes depends on the maximum time a network needs to confirm transactions.

6.3 Preventing counterfeit parts entering the supply chain

Regarding physical assets, the introduction of virtual identities, for example Hash IDs, can be seen as a new definition of ownership. The ownership of a physical asset can only be considered as complete if an entity owns both the physical asset and its virtual identity. Only this combination allows an entity to further perform valid actions with this asset, such as sending or merging it with other assets. This twofold security alone makes it considerably more difficult for counterfeit parts to enter the supply chain. As the practical implementation shows, the possession of the virtual identity in the form of the Hash ID is decisive for a valid continuation of actions. If, for example, a manufacturer receives a component on the distributed ledger whose virtual identity is invalid, the smart contract prevents the manufacturer from assembling this component with other valid components. Thus the manufacturer cannot proceed in creating an identity representing a valid final product or assembly. Due to the immutability of DLTs, the virtual merging of counterfeit parts can be considered impossible as long as the source code excludes such actions.

Even a scenario, with a valid Hash ID on an invalid physical component can be identified by checking the coincidence between the physical and virtual history. Since the entire history of an asset is stored on the distributed ledger, this history must match the circumstances in the physical world. For example if a customer wants to buy a certain part at a retailer, the address of the current owner of an asset must match the address of the retailer. In case Customer A receives an asset of Retailer B, but according to history on the distributed ledger Retailer C is the current owner of the asset, the data integrity is violated and the asset can be considered invalid and potentially counterfeit.

To identify assets that are exchanged with a counterfeit component in the course of the supply chain is most difficult. This is especially the case if the fraudster actually owned the valid virtual identity of an asset at some point of the supply chain. The actually valid physical component is separated from its virtual identity and it thus becomes worthless, since it can no longer meet the new definition of ownership if the framework is consistently enforced by the entire manufacturing supply chain. In addition, such fraud can only be carried out by a participant of the supply chain. This shows, that as long as there is a transition between the physical world and the virtual world, fraud can never be completely excluded. If, however,

this fraud occurs at any point, the history of each component can be traced back until the fraudster has been identified. If a fraudster's identity is successfully identified, it can be determined in a short time how far-reaching his participation in the supply chain is and which assemblies are potentially affected.

During the practical implementation the problem has occurred, that components can be assembled incorrectly or get damaged during the manufacturing process. The immutability of the DLT does not allow the deletion of virtual identities. Therefore, official methods of dealing with defective parts must be embedded in the source code of the smart contract, to prevent faulty parts from remaining in circulation and then leading to inconsistencies. Several solutions can be considered to prevent such problems. For example the affected Hash IDs can be marked as such by a mechanism blocking this asset for any further actions. This creates a blacklist for defective components. Another possibility is to send the incorrect identities to an address for which no private key is known. This creates a kind of 'virtual scrap yard', which makes it impossible for any entity to have access to those identities.

IBM states in their container tracking solution, that the adopting of the blockchain technology facilitates the tracking process to such an extent, that the effort to find a container can be reduced from 10 steps and five people to only one step and one person (Linnet & Wagner, 2018). Since the presented framework not only enables the tracking of individual goods, but also the tracing of every single component involved, similar time and resource cutbacks can also be regarded as realistic for complex manufacturing networks, especially when it comes to identifying parts which show inconsistencies. This applies to counterfeit parts as well as to unintentional occurrences of faulty components or batches, which can potentially trigger recalls. Even though the embedding of DLT cannot entirely exclude the possibility of counterfeit parts entering the supply chain, it can make a significant contribution when determining to what extent the inconsistent components and entities affect the entire supply chain.

6.4 Customer transparency

In the permissionless implementation, a user interface which is linked to a smart contract function is created to gather the information about the exact composition and history of an assembly. This function is not linked to a specific address and can therefore be considered as

a public function. This makes it possible for every network participant to check the current content of the smart contract. Since the network is permissionless any user can retrieve the current content of the smart contract and use it to check the composition and history of each component within the smart contract. Even if such a function is not accessible to the public the practical implementation shows, that users still have the possibility to evaluate the metadata of the distributed network. Metadata in the World Wide Web can be technically defined as data about data, “which can include activities, events, persons, places, structures, transactions, relationships, execution directions, and programmatic applications” (Greenberg, 2003, p. 1876). For example, in traditional chat applications these metadata are centrally stored and are only accessible to the application provider (Gerber et al., 2018). In DLTs, however, these metadata are accessible and transparent to all network participants. With regard to DLT-based smart contracts, metadata can provide information about the entities involved, the time of transaction, and the type of smart contract interaction. These metadata can serve as a basis of information for independent comparison portals on the distributed ledger in order to analyse transaction characteristics and can therefore increase the transparency for customers.

In the permissioned implementation the user interface is only accessible for a selected group of participants. Such an environment has the advantage that the networks can focus on increasing the supply chain visibility. Therefore the shared information can be more detailed and also contain information without any customer relevance but being of importance for the manufacturing network. Customer transparency can be created with the establishment of a supernode, which passes the network data to an application to which customers have access. However, this does not exclude the possibility of data being changed by the application. In addition, the customer only has access to a selected and filtered set of information. This results in restricted transparency for the customer because the customer has to trust the entity that provides the data.

As a result, the proposed framework distinguishes between two different levels of customer transparency. A permissionless platform gives the customer or other stakeholders outside the supply chain the opportunity to ensure complete transparency of all processes within the supply chain. On a permissioned platform, on the other hand, the customers can only obtain restricted transparency because they cannot actively participate at the distributed

ledger. Thus, on such platforms, the information must be provided by a node within the network. The verification based on the practical implementation of the framework shows that the high transparency provided on a permissionless platform can be generated via two channels: The transparency of the smart contract content and the transparency of the network. The validation of these two different possibilities for generating customer transparency shows that the smart contract-based solution proposed by the framework also makes possible a further new and complex solution. This new approach could be structured as follows:

As a first step the smart contract-based supply chain application can be encrypted and all its functions connected to entities, which are directly involved in the supply chain. All query functions of smart contract must be encrypted in such a way that only authorised participants can decrypt them and have access to the information stored in the smart contract. This construct allows the smart contract to contain detailed information about the supply chain and thus increase the visibility of the manufacturing network. As a next step, such a smart contract can be deployed on a permissionless distributed ledger network. Thus, the network participants who are not directly involved in the supply chain do not have the possibility to access the encrypted contents of the smart contract. However, they can still evaluate the metadata which are generated when authorised entities interact with the smart contract. Such new approach would result in a kind of ‘intranet’ on a permissionless distributed ledger, but with the metadata still available to the whole network. If, for example, an entity is involved in unethical machinations the network can analyse the metadata to determine the involvement of this entity in any supply chains. Therefore, this completely new approach would combine the advantages of the permissioned and permissionless approach based on the proposed framework. However, such a complex encrypted smart contract implementation is representing a new field of research and must be explored in a more detailed manner in further research activities.

7 Summary, conclusions, and recommendations

This chapter gives a brief overview of the work discussed in this thesis. The first section provides a summary of the individual chapters in terms of research content and approach. A description of how the research questions defined in Section 1.2 were approached and answered in the course of the research project follows. The chapter concludes with limitations and recommendations for future research projects.

7.1 Research summary

The thesis is divided into seven chapters, namely; (1) Introduction, (2) Literature review, (3) Research environment, (4) Framework development, (5) Practical implementation and dApp development, (6) Verification and validation of the framework, (7) Summary, conclusions and recommendations.

Chapter 1 introduces the research project and gives initial background information on distributed ledger technologies and manufacturing supply chains. Furthermore, the chapter describes the rationale of the research and states the problem and the research questions, as well as the research objectives. Finally, the chapter addresses the research methodology, the research design, and the outline of the thesis.

The literature review in Chapter 2 provides the reader with fundamental knowledge about manufacturing networks and distributed ledger technologies. In this context, the chapter explains important definitions and basic components that represent the knowledge base for the framework development. Finally, the chapter describes existing projects that are already testing distributed ledger technologies in conjunction with supply chains.

Chapter 3 illustrates the research environment and explains the constituents of the Reutlingen University and Stellenbosch University manufacturing network. Based on this environment, the chapter explains country-specific circumstances and the importance of research for the both Germany and South Africa.

In Chapter 4 the framework development takes place. The conceptual process consists of a network design stage, network planning stage, and an execution stage. This process is influenced by several physical complexity drivers and virtual complexity drivers. Taking these complexity drivers into account, the chapter proposes a framework with a possible procedure to increase the transparency through a tracking and tracing system by adopting the distributed ledger technology.

Chapter 5 describes the practical implementation of the framework and the programming of the decentralised application. The procedure corresponds to the procedure proposed by the framework. The example of a manufacturing supply chain described in Chapter 3 serves as implementation environment. A utility analysis results in Ethereum as a suitable platform for the practical implementation. Subsequently, the chapter explains the set-up of a permissioned Ethereum network and the main components of the smart contract's source code.

Chapter 6 verifies and validates the feasibility of the framework based on the results of the practical implementation. First, the chapter thematises the verification and validation of the smart contract's immunity and security. Subsequently, the chapter verifies and validates to what extent the practical blockchain-based implementation enables the tracking of manufactured goods. The chapter ends with the verification and validation of the customer transparency, which is enabled by the practical implementation of the framework.

7.2 Research conclusion and contribution

The results of the primary and secondary research questions are summarised below. The purpose is to present the key findings and conclusions to be drawn based on the research work.

Table 16: Primary research question 1

Primary research question 1	How can manufacturing networks adopt the distributed ledger technology to enable a transparent tracking of assembled goods and the auditability of all components they consist of?
------------------------------------	--

This elaboration proposes a framework to adopt the distributed ledger technology in a manufacturing network. The framework is divided into three main stages based on the general structure of supply chain management systems. First, in the network design stage, the framework suggests to identify and categorise all stakeholders involved in the manufacturing supply chain. The complexity of this stage depends on the globalisation of the supply base and the number of involved stakeholders. Additionally, this stage determines the extent to which customers have access to the network, which subsequently increases the complexity of the network planning stage. Finally, the network design stage results in an initial design of the network structure and all its associated nodes and their respective authorities.

Based on the network design the framework recommends in a network planning stage to define the respective asset compositions and a basic supply chain logic. The complexity of the products, the batch sizes, and the number of concurrent production processes influence the complexity of this stage. In order to track and trace a product by using a distributed ledger technology each asset must obtain a virtual identity on the distributed ledger. This virtual identity must have the same ownership and conversion characteristics as their physical counterpart. To meet these requirements the author has developed the approach of smart contract-based Hash IDs. The framework proposes the creation of models determining the dependencies for creating and linking the virtual identities logically. By means of this logic it is possible to derive fundamental requirements that are put to a platform necessary to enable a practical implementation of the framework. Depending on these requirements, the platform decision can result in both a permissionless platform and a permissioned platform.

In the final execution stage the framework describes the holistic implementation and integration of the smart contract-based decentralised application. Therefore, the smart contract logic models are implemented on the selected distributed ledger technology platform and they can subsequently be integrated into all production and logistics processes. An implementation based on a permissionless platform provides full transparency for all stakeholders, including the customer, while an implementation on a permissioned platform provides only restricted transparency for the customer. Distributed ledger technologies combined with smart contract-based virtual identities which are linked to their physical counterparts and which depend logically on their respective physical supply chain processes, make tracking of all manufactured goods possible. Additionally, distributed ledger technologies provide an immutable transaction history which enables the auditability of all manufactured goods as well as all components they consist of. To the knowledge of the author, Dr. Daniel Palm, and Dr. Louis Louw there no similar smart contract-based approach exists in research at the moment of submitting this thesis.

Table 17: Secondary research question 1

Secondary research question 1	How can the physical assets be linked to a virtual identity on the distributed ledger?
--------------------------------------	--

The virtual identity of an asset represents the asset on the distributed ledger. In order to ensure a consistent flow of physical and virtual processes, it must be guaranteed that the state of ownership and state of status of an asset in the physical world always matches the state of the virtual identity on the distributed ledger. To enable a clear identification of each asset, the framework recommends the establishment of a unique identification number which does not change over the entire life cycle of an asset. For this purpose the framework uses smart contract-based Hash IDs. A Hash ID is a unique identifier for a set of information on the distributed ledger representing an asset's virtual identity. The barcode technology and RFID technology are suitable for connecting the identifier of the virtual identities to the physical assets. As these identification numbers do not change over the entire life cycle, the engraving of parts is considered a suitable solution as well. In a practical implementation, QR codes have proved to be an effective solution for linking physical assets to their virtual identities.

Table 18: Secondary research question 2

Secondary research question 2	How can the distributed ledger technology reduce vulnerabilities of counterfeit parts entering the supply chain?
--------------------------------------	--

The introduction of virtual identities can be seen as a new definition of ownership. The ownership of a physical asset can only be considered as complete if an entity owns both the physical asset and its virtual identity. Only this combination allows an entity to perform further valid actions with this asset, such as sending it or merging it with other assets. The immutability of a smart contract source code prevents any actions with invalid virtual identities. At the same time, the state and ownership of a virtual identity can always be compared to the state and ownership of the physical asset. For example if a customer wants to buy a certain asset at a retailer, the address of the current owner of the asset on the distributed ledger must match the address of the retailer. If this is not the case, the asset is invalid and potentially counterfeit. It is most difficult to identify assets that are exchanged with a counterfeit component in the course of the supply chain. Especially if the fraudster actually owned the valid virtual identity of an asset at some point of the supply chain. Therefore, fraud can never be completely excluded, as long as there is a transition between the physical world and the virtual world.

Even though fraud cannot be completely excluded, the immutability of each asset's history allows a fast identification of all entities and assets involved after a fraud has successfully been identified. The supply chain mapped on a distributed ledger technology provides a complete information base to identify how far-reaching a problem is, including all potentially affected parts and all entities involved. This applies to both cases of fraud and unintentionally caused defects that can occur, for example, in production processes.

Table 19: Secondary research question 3

Secondary research question 3	What requirements does a platform have to meet in order to be eligible for a practical implementation?
--------------------------------------	--

The framework proposes an approach, which intends to create all virtual identities as well as their logical dependencies based on programmable smart contracts. Manufacturing supply

chains represent complex interrelationships. Therefore they require the platform to support a Turing Complete programming language to enable a mapping of these on a distributed ledger. At the time of writing this thesis the blockchain technology in particular is substituting the distributed ledger technology with the most advanced platforms supporting Turing Complete smart contracts.

The scalability of distributed ledger technology platforms is currently a limiting factor for decentralised applications with high transaction density. The platform and in particular its consensus mechanism must therefore have sufficient capacity in terms of scalability to support complex supply chain applications. The required scalability of the application depends on the properties of the respective manufacturing supply chain and can be predicted according to the defined logic models.

Depending on the objective of the supply chain application, the platform must be able to support either the permissionless use or the permissioned use. This depends mainly on the consensus mechanism of a platform. Consensus algorithms designed for the permissionless use can also work in permissioned networks. Consensus algorithms especially designed for permissioned environments, however, are not suitable for use in permissionless networks.

Table 20: Secondary research question 4

Secondary research question 4	How does an implementation differ from the use of a permissionless or permissioned network?
--------------------------------------	---

In a permissioned environment the whole capacity of the network is available for the decentralised tracking application. This leads to a considerably faster and more reliable confirmation of transactions. Tests showed that on a permissioned Ethereum network all transactions were confirmed within one second, while on the public Rinkeby Ethereum network the transaction confirmation took an average of 15.05 seconds. The shortest confirmation took 6 seconds and the longest confirmation 28 seconds. The speed of the transaction confirmation influences how detailed a tracking process can be designed. The time intervals between processes depends on the maximum time the network needs to confirm transactions in order to avoid conflicts and bottlenecks. Therefore, real-time tracking is only

possible if the network offers enough capacity in terms of scalability, which currently is only reliably possible on permissioned networks.

In addition, the implementation differs in the nature and scope of the shared information. Since in a permissionless environment any network participant has access to all data of the supply chain, the shared information consist likely on data with customer-relevance. In return, customers and other stakeholders who are not directly involved in the supply chain have the opportunity to gain unrestricted network transparency about the supply chain, either by a concrete evaluation of the smart contract contents or by an evaluation of the metadata of the network. In a permissioned network, however, only selected participants have access to the information. Accordingly, the shared information on permissioned networks can be more extensive and also include sensitive data. This can increase the supply chain visibility but at the expense of supply chain transparency for outside parties. For example with supernodes permissioned networks can still provide restricted transparency for customers. But that shared information no longer possesses the immutable properties of distributed ledger technologies.

Table 21: Secondary research question 5

Secondary research question 5	What does a practical implementation of a distributed ledger technology for tracking goods in manufacturing supply chains look like?
--------------------------------------	--

For this elaboration, Ethereum is selected as the implementation platform. For the implementation on a permissionless network the Ethereum Rinkeby test network is used. For the implementation on a permissioned network, an own Ethereum network is set up. For this network the full nodes are operated by desktop computers with windows as operating system, an Intel Core i7 (2.90 GHz) processor, and 16 GB RAM. The light nodes of the network each consist of a Raspberry Pi 3 with a processing speed of 1.40 GHz and 1 GB RAM. The smart contracts are programmed with Ethereum's programming language 'Solidity'. The graphic user interface is created with 'ReactJS'.

The operative process starts with the creation of virtual identities. In case the virtual identity refers to a physical asset, a QR code containing the Hash ID is attached to the physical asset. When an entity sends a physical asset to a new owner, the Hash ID of the asset must

simultaneously be sent to the blockchain address of the new owner. If an entity assembles two or more components in the physical world, the entity must select the respective Hash IDs of the components and trigger an assembling transaction to merge the two components on the blockchain. This will result in a new Hash ID, which must be attached to a newly assembled asset in the physical world. The source code of the smart contract ensures that only valid assets can be sent and merged by their valid owners on the blockchain. The whole transaction history of each asset is stored on the blockchain. All information about the current state and owner of every asset can be retrieved via the user interface.

7.3 Limitations and recommendations for further research

First, the basic limitations of the framework and the practical implementation are taken into account. Additionally, new further fields of research based on the results of this work are introduced.

The purpose of the framework is to propose a possible solution of how manufacturing networks can adopt the distributed ledger technology to solve current vulnerabilities in their supply chains. Since no comparable approach exists, the practical implementation was conducted in a relatively small research environment to prove the technical feasibility of such solution. Further research is necessary to investigate the feasibility of larger implementations, especially in terms of scalability, required storage and costs. Additionally it must be investigated, how this approach can be combined with existing tracking solutions such as GPS-tracking. From an economic point of view, it appears that the economic benefit for a company is only apparent if the introduction of such smart contract-based application generates added value. This can be expressed in different ways, such as reduced production costs or shortened production times. Therefore, research about specific products or industries that can drive the maximum value of such an approach represent an important next step for further research projects.

The framework is a platform independent approach and its practical implementation was based on existing platforms. As the validation shows, the scalability of distributed ledger technologies is severely limiting the realisation of decentralised tracking applications, particularly when trying to enable real-time tracking by adopting a distributed ledger

technology. Therefore, further research is necessary to invent more efficient consensus mechanisms in order to solve the current scalability problem of distributed ledger technologies.

In the proposed solution every asset receives a unique virtual identity, which is offering a promising foundation for further Internet-of-Things-based research activities. In this thesis, the virtual identities were only used for non-financial purposes. Nevertheless, distributed ledger technologies offer a suitable environment to connect virtual identities with financial applications. Further research is necessary to investigate how smart contract-based virtual identities can be combined with complex negotiation mechanisms and financial applications.

Based on the verification and validation of the proposed framework, there is also the possibility for a completely new approach of deploying a smart contract with encrypted content on a permissionless network. In this way, the participants of the manufacturing network could still store sensitive data in the smart contract resulting in a kind of ‘intranet’ on a permissionless distributed ledger, but with the metadata still available to the whole network. However, the feasibility of such a complex encrypted smart contract implementation represents a new field of research and must be explored in a more detailed manner in further research activities.

References

- Acharjya, D. P., & Geetha, M. K. (2017). *Internet of Things: Novel advances and envisioned applications. Studies in big data: Volume 25*. Cham: Springer.
- Adom, D., Hussein, E. K., & Joe, A.-A. (2018). Theoretical and conceptual framework: mandatory ingredients of a quality research. *International Journal of Scientific Research*, 7(1), 438-431.
- Anton Badev, & Matthew Chen (2014). Bitcoin: Technical Background and Data Analysis. *Finance and Economics Discussion Series, Divisions of Research & Statistics and Monetary Affairs, Federal Reserve Board, Washington, D.C.*
- Baliga, A. (2017). *Whitepaper - Understanding Blockchain Consensus Models: Persistent Systems*.
- Bano, S., Sonnino, A., Al-Bassam, M., Azouvi, S., McCorry, P., Meiklejohn, S., & Danezis, G. (2017). Consensus in the Age of Blockchains. *Cornell University*.
- Barclay (2019). Weiss Crypto Ratings Identifies Cryptos with Winning Combination of Adoption and Technology. Retrieved August 13, 2019, from Weiss Ratings LCC: <https://weissratings.com/articles/weiss-crypto-ratings-identifies-cryptos-with-winning-combination-of-adoption-and-technology>.
- Bashir, I. (2018). *Mastering blockchain: Distributed ledgers, decentralization, and smart contracts explained* (2. edition, fully revised and updated.). Birmingham: Packt Publishing.
- Baumann, K., & Supe, J. (2018). Blockchain als Treiber im modernen Blockchain als Treiber im modernen Supply Chain Management. Retrieved October 01, 2019, from BearingPoint: https://www.bearingpoint.com/files/Blockchain_im_SCM.pdf?download=0&itemId=552008.
- Bentov, I., Gabizon, A., & Mizrahi, A. (2016). Cryptocurrencies without Proof of Work. *Financial Cryptography and Data Security*, 142–157.
- Berentsen, A., & Schär, F. (2017). *Bitcoin, Blockchain und Kryptoassets*. Norderstedt: BoD - Books on Demand.
- Bertoni, G., Daemen, J., Hoffert, S., Peeters, M., van Asche, G., & van Keer, R. (2013). On 128-bit security. Retrieved August 09, 2019, from https://keccak.team/2013/on_128bit_security.html.

- Blakeman, M. (2019). Walmart and Sam's Club to Require Real-Time, End-to-End Food Traceability with Blockchain: Suppliers of Leafy Green Vegetables to Utilize Blockchain Technology. Retrieved August 20, 2019, from Walmart.com: https://corporate.walmart.com/media-library/document/leafy-greens-on-blockchain-press-release/_proxyDocument?id=00000166-0c4c-d96e-a3ff-8f7c09b50001.
- Blummer, T., Bohan, S., Bowman, M., Cachin, C., Gaski, N., George, N., . . . Yang, B. (2018). *An introduction to Hyperledger*: Creative Commons Attribution 4.0 International License. Retrieved from https://www.hyperledger.org/wp-content/uploads/2018/07/HL_Whitepaper_IntroductiontoHyperledger.pdf
- BMW Group (2019). Welcome to the BMW Group plant Rosslyn. Retrieved September 01, 2019, from BMW Group: <https://www.bmwgroup-plants.com/rosslyn/en.html>.
- Bosch (2019). Quality Bosch Parts Made in South Africa. Retrieved May 20, 2019, from Bosch: <https://www.bosch.africa/news-and-stories/bosch-brits-plant/>.
- Bott, J., & Milkau, U. (2016). Towards a framework for the evaluation and design of distributed ledger technologies in banking and payments. *Journal of Payments Strategy & Systems*, 10(2), 153–171.
- Bottcher, E. (2018). What I Talk About When I Talk About Platforms. Retrieved September 19, 2019, from <https://martinfowler.com/articles/talk-about-platforms.html>.
- Bozarth, C. C., Warsing, D. P., Flynn, B. B., & Flynn, E. J. (2009). The impact of supply chain complexity on manufacturing plant performance. *Journal of Operations Management*, 27(1), 78–93.
- Brandon-Jones, E., Squire, B., Autry, C., & Pertersen, K. (2014). A contingent resource-based perspective of supply chain resilience and robustness. *Journal of Supply Chain Management*. (50), 55–73.
- Brennan, C., & Lunn, W. (2016). Blockchain - The Trust Disrupter. *Connection Series*.
- Bruehl, V. (2017). Bitcoins, Blockchain und Distributed Ledgers. *Wirtschaftsdienst*, 97(2), 135–142.
- Buterin, V. (2013). *Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform*.
- Buterin, V. (2014a). DAOs, DACs, DAs and More: An Incomplete Terminology Guide. Retrieved October 20, 2019, from <https://blog.ethereum.org/2014/05/06/daos-dacs-das-and-more-an-incomplete-terminology-guide/>.
- Buterin, V. (2014b). On Stake. Retrieved October 20, 2019, from <https://blog.ethereum.org/2014/07/05/stake/>.
- Buterin, V. (2015). On Public and Private Blockchains. Retrieved October 20, 2019, from <https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/>.

- Camarinha-Matos, L., & Afsarmanesh, H. (2006). *Collaborative Networks - Value creation in a knowledge society*. China: Springer.
- Castro, M., & Liskov, B. (1999). Practical Byzantine Fault Tolerance. *Operating Systems Design and Implementation*, 173–186.
- Chauhan, A., Malviya, O. P., Verma, M., & Mor, S. T. (2018). Blockchain and Scalability. *IEEE International Conference on Software Quality, Reliability and Security Companion*.
- Chen, S., Zhang, Q., & Zhou, Y.-P. (2018). Impact of Supply Chain Transparency on Sustainability under NGO Scrutiny. *Production and Operations Management*, 63(9).
- Chohan, U. W. (2019). Decentralized Applications (dApps): A Short Primer. *University of New South Wales (UNSW)*.
- Christopher, M. (2016). *Logistics & supply chain management. Always learning*. Harlow, England, New York: Pearson Education.
- Christopher, M., & Lee, H. (2004). Mitigating supply chain risk through improved confidence. *International Journal of Physical Distribution & Logistics Management*, 34(5).
- Churyumov, A. (2016). Byteball: A Decentralized System for Storage and Transfer of Value. Retrieved September 12, 2019, from <https://obyte.org/Byteball.pdf>.
- Collier, Z. A., Hassler, M. L., Lambert, J. H., DiMase, D., & Linkov, I. (2019). Supply Chains. In A. Kott & I. Linkov (Eds.), *Cyber Resilience of Systems and Networks* (Vol. 2, pp. 447–462). Cham: Springer International Publishing.
- Cuccuru, P. (2017). Beyond bitcoin: an early overview on smart contracts. *International Journal of Law and Information Technology*, 25(3), 179–195.
- Daimler (2019). East London, Mercedes-Benz Manufacturing South Africa Ltd. Retrieved September 01, 2019, from Daimler: <https://www.daimler.com/career/about-us/locations/location-detail-page-5096.html>.
- Dang, Q. (2012). Recommendation for Applications Using Approved Hash Algorithms: NIST Special Publication 800-107 - Revision 1. *National Institute of Standards and Technology*.
- Dannen, C. (2017). *Introducing Ethereum and Solidity: Foundations of cryptocurrency and blockchain programming for beginners*. New York: Apress.
- David Im, D. K. The Blockchain Trilemma - The technology trade-offs among the security, decentralization, and scalability of blockchain. *University of Pennsylvania*, 2018.
- De Beers Group (2018). Alrosa pilots Tracr platform. Retrieved October 19, 2019, from <https://www.debeersgroup.com/media/company-news/2018/alrosa-pilots-tracr-platform>.

- Dhillon, V., Metcalf, D., & Hooper, M. (2017). *Blockchain Enabled Applications*. Berkeley, CA: Apress.
- Digiconomist (2019). Ethereum Energy Consumption Index (beta). Retrieved October 17, 2019, from <https://digiconomist.net/ethereum-energy-consumption>.
- DiMase, D., Collier, Z. A., Carlson, J., Gray, R. B., & Linkov, I. (2016). Traceability and Risk Analysis Strategies for Addressing Counterfeit Electronics in Supply Chains for Complex Systems. *Risk Analysis : an Official Publication of the Society for Risk Analysis*, 36(10), 1834–1843.
- Dinh, T. T. A., Liu, R., Zhang, M., Chen, G., Ooi, B. C., & Wang, J. (2018). Untangling Blockchain: A Data Processing View of Blockchain Systems. *IEEE Transactions on Knowledge and Data Engineering*, 30(7), 1366–1385.
- Dinh, T. T. A., Wang, J., Chen, G., Liu, R., Ooi, B. C., & Tan, K.-L. (2017). BLOCKBENCH: A Framework for Analyzing Private Blockchains. *ACM International Conference on Management of Data*, 1085–1100.
- Domon, K. (2018). *An Economic Analysis of Intellectual Property Rights Infringement: Field Studies in Developing Countries*. *Palgrave Studies in Institutions, Economics and Law*. Cham: Springer International Publishing.
- Doorey, D. (2011). The Transparent Supply Chain: from Resistance to Implementation at Nike and Levi-Strauss. *Journal of Business Ethics*. (103), 587–603.
- Duckworth, N. (2018). Supply Chain Visibility and Transparency: How Everybody Wins. Retrieved October 20, 2019, from [tdwi: https://tdwi.org/articles/2018/07/10/data-all-supply-chain-visibility-and-transparency.aspx](https://tdwi.org/articles/2018/07/10/data-all-supply-chain-visibility-and-transparency.aspx).
- Elrom, E. (2019). *The Blockchain developer: A practical guide for designing, implementing, publishing, testing, and securing distributed Blockchain-based projects*. Berkeley, California: Apress.
- EOS Authority (2019). Block Producer Elections (Schedules). Retrieved September 19, 2019, from https://eosauthority.com/producers_schedules?network=eos.
- EOS Whitepaper (2018). EOS.IO Technical White Paper v2. Retrieved September 19, 2019, from <https://github.com/EOSIO/Documentation/>.
- Eosio.cdt (2019). EOS Developer. Retrieved September 19, 2019, from <https://eosio.github.io/eosio.cdt/latest>.
- Ernst, R., & Haar, J. (2019). *Globalization, competitiveness, and governability: The three disruptive forces of business in the 21st century*. Cham, Switzerland: Palgrave Macmillan.
- Eskandarpour, M., Dejax, P., Miemczyk, J., & Péton, O. (2015). Sustainable supply chain network design: An optimization-oriented review. *Omega*, 54, 11–32.

- Eßig, M., Hülsmann, M., Kern, E.-M., & Klein-Schmeink, S. (2013). *Supply chain safety management: Security and robustness in logistics. Lecture Notes in Logistics*. Berlin, Heidelberg: Springer.
- Ethereum Revision (2019). Contract ABI Specification. Retrieved August 10, 2019, from Ethereum Revision: <https://solidity.readthedocs.io/en/v0.5.3/abi-spec.html>.
- Fadilpasic, S. (2019). EOS Gets New Update for Better Transaction Speed. Retrieved August 19, 2019, from <https://cryptonews.com/news/eosio-gets-new-update-for-better-transaction-speed-3261.htm>.
- Fallenbeck, N., & Eckert, C. (Eds.). (2017). *IT-Sicherheit und Cloud Computing*: Springer.
- Feng, T. (2016). An agri-food supply chain traceability system for China based on RFID & blockchain technology. *13th International Conference on Service Systems and Service Management*, 1–6.
- Francis, V. (2008). Supply chain visibility: lost in translation? *Supply Chain Management: An international Journal*. (3), 180–184.
- Frost, & Sullivan. (2019). *Die perfekte Versorgungskette durch Digitalisierung*. Frankfurt: IBM.
- Garcia, D. J., & You, F. (2015). Supply chain design and optimization: Challenges and opportunities. *Computers & Chemical Engineering*, 81, 153–170.
- Gartner (2015). Gartner's 2015 Hype Cycle for Emerging Technologies Identifies the Computing Innovations That Organizations Should Monitor. Retrieved October 21, 2019, from Gartner: <https://www.gartner.com/en/newsroom/press-releases/2015-08-18-gartners-2015-hype-cycle-for-emerging-technologies-identifies-the-computing-innovations-that-organizations-should-monitor>.
- Gartner (2016). Gartner's 2016 Hype Cycle for Emerging Technologies Identifies Three Key Trends That Organizations Must Track to Gain Competitive Advantage. Retrieved October 21, 2019, from Gartner: <https://www.gartner.com/en/newsroom/press-releases/2016-08-16-gartners-2016-hype-cycle-for-emerging-technologies-identifies-three-key-trends-that-organizations-must-track-to-gain-competitive-advantage>.
- Gartner (2018). 5 Trends Emerge in the Gartner Hype Cycle for Emerging Technologies, 2018. Retrieved October 21, 2019, from Gartner: <https://www.gartner.com/smarterwithgartner/5-trends-emerge-in-gartner-hype-cycle-for-emerging-technologies-2018/>.
- Gentemann, L. (2019). Blockchain in Deutschland – Einsatz, Potenziale, Herausforderungen. Retrieved October 10, 2019, from www.bitkom.org.
- Gerber, N., Zimmermann, V., Henhapl, B., Emeroez, Volkamer, M., & Hilt, T. (2018). Nutzerwahrnehmung der Ende-zu-Ende-Verschlüsselung in WhatsApp. *Datenschutz und Datensicherheit*, 11, 680–685.

- Grant, C., & Osanloo, A. (2014). Understanding, selecting, and integrating a theoretical framework in dissertation research: creating the blueprint for your “house”. *Administrative Issues Journal Education Practice and Research*.
- Greenberg, J. (2003). Metadata and the World Wide Web. *Encyclopedia of Library and Information Science*, 1876–1888.
- Grinberg, R. (2011). Bitcoin: An Innovative Alternative Digital Currency. *Hastings Science & Technology Law Journal*. (4), 159–208.
- Grossman, N. (2015). The Blockchain as verified public timestamps. Retrieved October 21, 2019, from <https://www.nickgrossman.is/2015/the-blockchain-as-time/>.
- GS1 Germany (2018). Putting blockchain to the test - Key findings from Germany’s largest cross-company pilot project. Retrieved October 21, 2019, from https://www.gs1-germany.de/fileadmin/gsl/basis_informationen/putting_blockchain_to_the_test.pdf.
- Guin, U., Huang, K., DiMase, D., Carulli, J. M., Tehranipoor, M., & Makris, Y. (2014). Counterfeit Integrated Circuits: A Rising Threat in the Global Semiconductor Supply Chain. *Proceedings of the IEEE*, 102(8), 1207–1228.
- Hahn, C., & Wons, A. (2018). *Initial Coin Offering (ICO): Unternehmensfinanzierung auf Basis der Blockchain-Technologie. essentials*. Wiesbaden: Springer Gabler.
- Hoepman, J.-H. (2007). Distributed Double Spending Prevention. *International Workshop on Security Protocols*, 15, 152–165.
- Hu, S. J., Zhu, X., Wang, H. [H.], & Koren, Y. (2008). Product variety and manufacturing complexity in assembly systems and supply chains. *CIRP Annals*, 57(1), 45–48.
- Hu, Y., Liyanage, M., Mansoor, A., Thilakarathna, K., Jourjon, G., & Seneviratne, A. (2018, September 26). *Blockchain-based Smart Contracts - Applications and Challenges*.
- Iansiti, M., & Lakhani, K. R. (2017). *The Truth About Blockchain*: Harvard Business Review.
- IBM Research Editorial Staff (2018). Behind the Architecture of Hyperledger Fabric. Retrieved August 19, 2019, from <https://www.ibm.com/blogs/research/2018/02/architecture-hyperledger-fabric/>.
- IOTA (2019). Global Trade & Supply Chains. Retrieved 25.09.10, from IOTA.org: <https://www.iota.org/verticals/global-trade-supply-chains>.
- Jabareen, Y. (2009). Building a Conceptual Framework: Philosophy, Definitions, and Procedure. *International Journal of Qualitative Methods*, 8(4), 49–62
- Jakiman (2017). PIVX Zerocoin (zPIV) Technical Paper. Retrieved October 21, 2019, from https://pivx.org/de/white-papers_de_white-paper/.

- Kaplanov, N. M. (2012). Nerdy Money: Bitcoin, the Private Digital Currency, and the Case Against its Regulation. *Consumer L. Rev.* (25), 111–174.
- Khan, S. A. R., & Yu, Z. (2019). *Strategic supply chain management. EAI/Springer Innovations in Communication and Computing*. Cham: Springer.
- Kiayias, A., Koutsoupias, E., Kyropoulou, M., & Tselekounis, Y. (2016). Blockchain Mining Games. *ERC Advanced Grant 321171*.
- Kiayias, A., Russell, A., David, B., & Oliynykov, R. (2017). Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. Retrieved October 21, 2019, from <https://www.cardano.org/en/academic-papers/>.
- King, S., & Nadal, S. (2012). PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. Retrieved October 21, 2019, from https://www.researchgate.net/publication/265116876_PPCoin_Peer-to-Peer_Crypto-Currency_with_Proof-of-Stake.
- Kovacs, G., & Paganelli, P. (2003). A planning and management infrastructure for large, complex, distributed projects—beyond ERP and SCM. *Computers in Industry.* (51), 165–183.
- Kuehnappel, J. B. (2019). *Nutzwertanalysen in Marketing und Vertrieb* (2. Aufl. 2019). *essentials*. Wiesbaden: Springer Fachmedien Wiesbaden.
- Lambert, D. M., Cooper, M. C., & Pagh, J. D. (1998). Supply Chain Management: Implementation Issues and Research Opportunities. *The International Journal of Logistics Management*, 9(2), 1–20.
- Lambert, N., & Tolmay, A. S. (2017). Performance Of South African Automotive Exports Under The African Growth And Opportunity Act From 2001 To 2015. *International Business & Economics Research Journal*, 16(2), 131–142.
- Lambrecht, N. (2019). *Automotive Export Manual 2019*: AIEC.
- Leavy, P. (2017). *Research Design: Quantitative, qualitative, mixed methods, arts-based, and community-based participatory research approaches*. New York: The Guilford Press.
- Lee, D. K. C., & Deng, R. H. (Eds.). (2018). *Handbook of blockchain, digital finance, and inclusion. Chinatech, mobile security, and distributed ledger*. London: Academic Press.
- Lehikoinen, T. (2018). This summer, fishing in Finland means food traceability on the menu. Retrieved October 21, 2019, from <https://www.ibm.com/blogs/blockchain/2018/07/this-summer-fishing-in-finland-means-food-traceability-on-the-menu/>.
- Lehmacher, W. (2016). *Globale Supply Chain: Technischer Fortschritt, Transformation und Circular Economy*. Wiesbaden: Springer Gabler.
- LeMahieu, C. (2017). Nano: A Feeless Distributed Cryptocurrency Network. Retrieved October 22, 2019, from <https://nano.org/en/whitepaper>.

- Linich, D. (2014). *The path to supply chain transparency: A practical guide to defining, understanding, and building supply chain transparency in a global economy*: Deloitte University Press.
- Linnet, E., & Wagner, S. (2018). Maersk and IBM Introduce TradeLens Blockchain Shipping Solution. Retrieved October 22, 2019, from <https://newsroom.ibm.com/2018-08-09-Maersk-and-IBM-Introduce-TradeLens-Blockchain-Shipping-Solution>.
- Liu, X., Wang, W., Niyato, D., Zhao, N., & Wang, P. (2018). Evolutionary Game for Mining Pool Selection in Blockchain Networks. *IEEE Wireless Communications Letters*, 7(5), 760–763.
- Luu, L., Chu, D.-H., Olickel, H., Saxena, P., & Hobor, A. (2016). Making Smart Contracts Smarter. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications*, 254–269.
- Machado, S. M., Paiva, E. L., & da Silva, E. M. (2018). Counterfeiting: addressing mitigation and resilience in supply chains. *International Journal of Physical Distribution & Logistics Management*, 48(2), 139–163.
- Mauil, R., Godsiff, P., Mulligan, C., Brown, A., & Kewell, B. (2017). Distributed ledger technology: Applications and implications. *Strategic Change*, 26(5), 481–489.
- Mehar, M. I., Shier, C. L., Giambattista, A., Gong, E., Fletcher, G., Sanayhie, R., . . . Laskowski, M. (2019). Understanding a Revolutionary and Flawed Grand Experiment in Blockchain. *Journal of Cases on Information Technology*, 21(1), 19–32.
- Mills, D., Wang, K., Malone, B., Ravi, A., Marquardt, J., Chen, C., . . . Baird, M. (2016). Distributed Ledger Technology in Payments, Clearing, and Settlement. *Finance and Economics Discussion Series*, 2016(095).
- Mingxiao, D., Xiaofeng, M., Zhe, Z., Xiangwei, W., & Qijun, C. (2017). A Review on Consensus Algorithm of Blockchain. *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2567-2572.
- Mourtzis, D., & Doukas, M. (2012). Decentralized Manufacturing Systems Review: Challenges and Outlook. *Logistics Research*. (5), 113–121.
- Mourtzis, D., Doukas, M., & Psarommatis, F. (2015). A toolbox for the design, planning and operation of manufacturing networks in a mass customisation environment. *Journal of Manufacturing Systems*. (36), 274–286.
- Mueller, C. (2019). IOTA's Scalability Solution. Retrieved August 29, 2019, from <https://iota-news.com/iotas-scalability-solution/>.
- Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*.
- National Research Council (U.S.). (2000). *Surviving supply chain integration: Strategies for small manufacturers*. Washington, D.C: National Academy Press.

- Naude, M., & Badenhorst-Weiss, J. (2011). Supply chain management problems at South African automotive component manufacturers. *Southern African Business Review*, (15), 70–99.
- Nebilo Whitepaper (2017). Nebilo - Next Generation Enterprise Blockchain Solutions. Retrieved October 22, 2019, from <https://nebl.io/wp-content/uploads/2019/06/NeblioWhitepaper.pdf>.
- NEO Whitepaper (2019). Neo Whitepaper - A distributed network for the Smart Economy. Retrieved September 29, 2019, from <https://docs.neo.org/docs/en-us/basic/whitepaper.html>.
- New, S. (2010). The Transparent Supply Chain. Retrieved October 22, 2019, from Harvard Business Review: <https://hbr.org/2010/10/the-transparent-supply-chain>.
- Nguyen, G.-T., & Kim, K. (2018). A Survey about Consensus Algorithms Used in Blockchain. *Journal of Information Processing Systems*, 14(1), 101–128.
- Njui, J. (2018). Vitalik: Ethereum (ETH) Can Scale to 500 tps Using ZCash's ZK-SNARKs. Retrieved September 19, 2019, from <https://ethereumworldnews.com/vitalik-ethereum-eth-can-scale-to-500-tps-using-zcashs-zk-snarks/>.
- OEC (2017a). Observatory of Economic Complexity - Germany. Retrieved October 01, 2019, from <https://oec.world/en/profile/country/deu/>.
- OEC (2017b). Observatory of Economic Complexity - South Africa. Retrieved October 01, 2019, from <https://oec.world/en/profile/country/zaf/>.
- Olleros, F., & Zhegu, M. (2016). *Research Handbook on Digital Transformations*: Edward Elgar Publishing.
- Osipktov, I., Vasserman, E., & Hopper, N. (2007). Combating Double-Spending Using Cooperative P2P Systems. *Conference on Distributed Computing Systems*, 27.
- Pahl, C., Ioini, N. E., & Helmer, S. (2018). A Decision Framework for Blockchain Platforms for IoT and Edge Computing. *Institute for Systems and Technologies of Information*.
- Panetta, K. (2018). Blockchain, quantum computing, augmented analytics and artificial intelligence will drive disruption and new business models. Retrieved October 22, 2019, from Gartner: <https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2019/>.
- Pecht, M. (2013). The Counterfeit Electronics Problem. *Open Journal of Social Sciences*, 01(07), 12–16.
- Peresson, S. (2019). Counterfeit automotive parts increasingly putting consumer safety at risk. Retrieved October 22, 2019, from <https://www.lexology.com/library/detail.aspx?g=9ecb7809-e01c-4710-ae00-10ec3c1f7e1b>.

- Perez-Sola, C., Delgado-Segura, S., Navarro-Arribas, G., & Herrera-Joancomarti, J. (2018). Double-spending Prevention for Bitcoin zero-confirmation transactions. *International Journal of Information Security*, 18, 451–463.
- Popov, S. (2018a). On the Tangle, White Papers, Proofs, Airplanes, and Local Modifiers. Retrieved September 25, 2019, from <https://blog.iota.org/on-the-tangle-white-papers-proofs-airplanes-and-local-modifiers-44683aff8fea>.
- Popov, S. (2018b). The Tangle. Retrieved September 25, 2019, from https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf.
- Primavera de Filippi, Benedikt Schuppli, Constance Choi, Carla Reyes, et al., & Don Tapscott (2019). Regulatory Framework for Token Sales: An Overview of Relevant Laws and Regulations in Different Jurisdictions Coalition of Automated Legal Applications.
- Pugh, B. (2018). Stress Testing The RaiBlocks Network: Part II. Retrieved September 16, 2019, from <https://medium.com/@bnp117/stress-testing-the-raiblocks-network-part-ii-def83653b21f>.
- Pundir, A. K., Jagannath, J. D., & Ganapathy, L. (2019). Improving supply chain visibility using IoT-Internet of things. *IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, 156–162.
- Qubic.iota.org (2019). What is Qubic? Retrieved September 25, 2019, from <https://qubic.iota.org/intro>.
- Reyna, A., Martín, C., Chen, J., Soler, E., & Díaz, M. (2018). On blockchain and its integration with IoT. Challenges and opportunities. *Future Generation Computer Systems*, 88, 173–190.
- Rolland, F. D. (2003). *Datenbanksysteme im Klartext. Im Klartext*. München: Pearson Studium.
- Rosenberger, P. (2018). *Bitcoin und Blockchain*. Berlin, Heidelberg: Springer Berlin Heidelberg.
- Rudberg, M., & Olhager, J. (2003). Manufacturing networks and supply chains: an operations strategy perspective. *Omega - The International Journal of Management Science*. (31), 29–39.
- Sajana, P., Sindhu, M., & Sethumadhavan, M. (2019). On Blockchain Applications: Hyperledger Fabric And Ethereum. *International Journal of Pure and Applied Mathematics*, 118(18), 2965–2970.
- SCDigest, & jda (2016). SCDigest Supply Chain Digitization Benchmark Survey. Retrieved October 23, 2019, from http://www.scdigest.com/assets/reps/Supply_Chain_Digitization_2016_Survey_Data.pdf.

- Schiener, D. (2017). A Primer on IOTA. Retrieved September 16, 2019, from <https://blog.iota.org/a-primer-on-iota-with-presentation-e0a6eb2cc621>.
- Schlink, H. (2019). *Wirtschaftlichkeitsrechnung für Ingenieure: Grundlagen für die Entwicklung technischer Produkte*. Wiesbaden: Springer Fachmedien Wiesbaden.
- Schwarz-Kocher, M., Krzywdzinski, M., & Korflür, I. (Eds.). (2019). *Study: Nr. 409. Standortperspektiven in der Automobilzulieferindustrie: Die Situation in Deutschland und Mitteleuropa unter dem Druck veränderter globaler Wertschöpfungsstrukturen*. Düsseldorf: Hans-Böckler-Stiftung.
- Serdarasan, S. (2013). A review of supply chain complexity drivers. *Computers & Industrial Engineering*. (66), 533–540.
- Singhal, B., Dhameja, G., & Panda, P. S. (2018). *Beginning Blockchain: A Beginner's Guide to Building Blockchain Solutions*. Berkeley, CA: Apress.
- Sorge, C., & Krohn-Grimberghe, A. (2012). Bitcoin: Eine erste Einordnung. *DuD - Datenschutz und Datensicherheit*, 479–484.
- Statista (2017). Wie nehmen Sie Produkte wahr, auf denen steht ‚Made in ...‘? Retrieved September 18, 2019, from Statista.com: <https://de.statista.com/statistik/daten/studie/676530/umfrage/made-in-country-index-gesamtranking-2017/>.
- Stoica, I., Morris, R., Liben-Nowell, D., Karger, D., Kaashoek, F., Dabek, F., & Balakrishnan, H. *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications*.
- Sukhwani, H., Martinez, J., Chang, X., Trivedi, K., & Rindos, A. (2017). Performance Modeling of PBFT Consensus Process for Permissioned Blockchain Network (Hyperledger Fabric). *IEEE Symposium on Reliable Distributed Systems*. (36), 253–255.
- Szabo, N. (1994). Smart Contracts. Retrieved August 01, 2019, from <http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html>.
- Szabo, N. (1997). Formalizing and Securing Relationships on Public Networks. *First Monday*, 2(9). <https://doi.org/10.5210/fm.v2i9.548>
- TNW (2019). Stack Overflow Analysis: The top 10 programming languages in blockchain. Retrieved August 19, 2019, from thenextweb.com/hardfork/2019/05/24/javascript-programming-java-cryptocurrency/.
- Tosovic, V. (2016). Der DAO-Hack – und die Konsequenzen für die Blockchain. In D. Burgwinkel (Ed.), *Blockchain Technology* (pp. 159–166). Berlin, Boston: De Gruyter.

- Trillo, M. (2013). Stress Test Prepares VisaNet for the Most Wonderful Time of the Year. Retrieved October 23, 2019, from Visa Viewpoints: <https://www.visa.com/blogarchives/us/2013/10/10/stress-test-prepares-visanet-for-the-most-wonderful-time-of-the-year/index.html>.
- Truby, J. (2018). Decarbonizing Bitcoin: Law and policy choices for reducing the energy consumption of Blockchain technologies and digital currencies. *Energy Research & Social Science*, 44, 399–410.
- Ueda, K., Takenaka, T., Vancza, J., & Monostori, L. (2009). Value creation and decision-making in sustainable society. *CIRP Annals - Manufacturing Technology*. (58), 681–700.
- Valenta, M., & Sandner, P. (2017). Comparison of Ethereum, Hyperledger Fabric and Corda. *FSBC Working Paper*.
- Viswanathan, S., & Shah, A. (2018). The Scalability Trilemma in Blockchain. Retrieved September 19, 2019, from https://medium.com/@aakash_13214/the-scalability-trilemma-in-blockchain-75fb57f646df.
- Vogel-Heuser, B., Bauernhansl, T., & Hompel, M. ten. (2017). *Handbuch Industrie 4.0 Bd.4*. Berlin, Heidelberg: Springer Berlin Heidelberg.
- Volkswagen (2019). Volkswagen Group South Africa, Ltd. Uitenhage. Retrieved June 03, 2019, from Volkswagen: <https://www.volkswagen-newsroom.com/en/press-releases/volkswagen-group-south-africa-ltd-uitenhage-1296>.
- Vujicic, D., Jagodic, D., & Randic, S. (2018). Blockchain technology, bitcoin, and Ethereum: A brief overview, 1–6.
- Wang, W., Hoang, D. T., Hu, P., Xiong, Z., Niyato, D., Wang, P., . . . Kim, D. I. (2019). A Survey on Consensus Mechanisms and Mining Strategy Management in Blockchain Networks. *IEEE Access*, 7, 22328–22370.
- Watanabe, H., Fujimura, S., Nakadaira, A., Miyazaki, Y., Akutsu, A., & Kishigami, J. (2016). Blockchain contract: Securing a blockchain applied to smart contracts. *IEEE International Conference on Consumer Electronics (ICCE)*, 467–468.
- Werner, H. (2017). *Supply Chain Management: Grundlagen, Strategien, Instrumente und Controlling. Lehrbuch*. Wiesbaden: Springer Gabler.
- Wood, G. (2014). Ethereum: A Secure Decentralised Generalised Transaction Ledger - EIP-150 Revision.
- Wood, G. (2019). *Ethereum: A Secure Decentralised Generalised Transaction Ledger - Byzantium Version 3e36772-2019-05-12*.
- Workman, D. (2019a). Germany's Top 10 Exports. Retrieved 01.10.19, from worldstopexports.com: <http://www.worldstopexports.com/germanys-top-10-exports/>.

Workman, D. (2019b). Top South African Trading Partners. Retrieved October 15, 2019, from <http://www.worldstopexports.com/top-south-african-import-partners/>.

Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. [Huaimin] (2017). An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. *IEEE 6th International Congress on Big Data*, 557–564.

Zorzo, A. F., Nunes, H. C., Lunardi, R. C., Michelin, R. A., & Kanhere, S. S. (2018). Dependable IoT Using Blockchain-Based Technology. *Latin-American Symposium on Dependable Computing*.

Appendix A: Smart contract

A1: Smart contract Solidity source code

```
1. pragma solidity >=0.5.8;
2. contract SupplyChain{
3.
4.     address certifier;
5.     address supplierA;
6.     address supplierB;
7.     address manufacturer;
8.
9.     mapping(bytes32 => Certificate) public govtToManufacturerHistory;
10.    mapping(bytes32 => Certificate) public govtToSupplierAHistory;
11.    mapping(bytes32 => Certificate) public govtToSupplierBHistory;
12.    mapping(bytes32 => Certificate) public supplierAToManufacturerHistory;
13.    mapping(bytes32 => Certificate) public supplierBToManufacturerHistory;
14.    mapping(bytes32 => Certificate) public productsOwnedByManufacturer;
15.
16.    mapping(bytes32 => Supplier) public supplierDetails;
17.    mapping(bytes32 => Manufacturer) public manufacturerDetails;
18.    mapping(bytes32 => Product) public productDetails;
19.
20.    bytes32[] govtToManufacturerCertificates;
21.    bytes32[] govtToSupplierACertificates;
22.    bytes32[] govtToSupplierBCertificates;
23.    bytes32[] supplierBToManufacturerCertificates;
24.    bytes32[] supplierAToManufacturerCertificates;
25.    bytes32[] productCertificates;
26.
27.
28.    struct Supplier{
29.        string name;
30.        string nameOfCertificates;
31.        string date;
32.        bytes32 certificateHash;
33.    }
34.
35.    struct Manufacturer{
36.        string name;
37.        string nameOfCertificates;
38.        string date;
39.        bytes32 certificateHash;
40.    }
41.
42.
43.    struct Product{
44.        string nameOfProducer;
45.        string typeOfProduct;
46.        string serialId;
47.        string date;
48.        bytes32 certificateHash_p;
49.        bytes32 certificateHash_sup_a;
50.        bytes32 certificateHash_sup_b;
51.        bytes32 certificateHash_govt;
52.    }
53.
54.
```

```

55.     struct Certificate{
56.         bytes32 hash;
57.         bool used;
58.     }
59.
60.     constructor(address _addressA, address _addressB, address _addressM, address _c
certifier) public{
61.         supplierA = _addressA;
62.         supplierB = _addressB;
63.         manufacturer = _addressM;
64.         government = _certifier;
65.     }
66.
67.
68.     //used by certifier
69.     function createAndSendCertificate(address _address, string memory _name, string
memory _nameOfCertificate, string memory _date, uint256 _role) public{
70.         if(_role == 0){
71.             if(_address == supplierA){
72.                 bytes32 certHash = keccak256(abi.encodePacked(_name,_nameOfCertific
ate,_date,"A",block.timestamp));
73.                 Certificate memory cert = Certificate(certHash,false);
74.                 govtToSupplierAHistory[certHash] = cert;
75.                 govtToSupplierACertificates.push(certHash);
76.                 supplierDetails[certHash] = Supplier(_name,_nameOfCertificate,_date
,certHash);
77.             }else if(_address == supplierB){
78.                 bytes32 certHash = keccak256(abi.encodePacked(_name,_nameOfCertific
ate,_date,"B",block.timestamp));
79.                 Certificate memory cert = Certificate(certHash,false);
80.                 govtToSupplierBHistory[certHash] = cert;
81.                 govtToSupplierBCertificates.push(certHash);
82.                 supplierDetails[certHash] = Supplier(_name,_nameOfCertificate,_date
,certHash);
83.             }
84.
85.         }else{
86.             require(_address == manufacturer,"Manufacturer address not correct");
87.             bytes32 certHash = keccak256(abi.encodePacked(_name,_nameOfCertificate,
_date,"M"));
88.             Certificate memory cert = Certificate(certHash,false);
89.             govtToManufacturerHistory[certHash] = cert;
90.             govtToManufacturerCertificates.push(certHash);
91.             manufacturerDetails[certHash] = Manufacturer(_name,_nameOfCertificate,
date,certHash);
92.
93.         }
94.     }
95. }
96.
97.
98.     //used by suppliers
99.     function createCertificateForManufacturer(address _address,string memory _nameO
fProducer, string memory _typeOfProduct,
100.         string memory _serialId, string memory _date,bytes32 _certHash) public
{
101.         bytes32 certHash = keccak256(abi.encodePacked(_nameOfProducer,_typ
eOfProduct,_serialId,_date,block.timestamp));
102.         if(_address == supplierA){
103.             govtToSupplierAHistory[_certHash].used = true;
104.             supplierAToManufacturerCertificates.push(certHash);
105.         }else if(_address == supplierB){
106.             govtToSupplierBHistory[_certHash].used = true;
107.             supplierBToManufacturerCertificates.push(certHash);
108.         }

```

```

109.     }
110.
111.     //used by suppliers
112.     function sendCertificateToManufacturer(address _address,bytes32 _certH
ash) public {
113.         if(_address == supplierA){
114.             supplierAToManufacturerHistory[_certHash]=Certificate(_certHas
h,false);
115.
116.         }else if(_address == supplierB){
117.             supplierBToManufacturerHistory[_certHash]=Certificate(_certHas
h,false);
118.
119.         }
120.     }
121.
122.
123.     //used by manufacturer
124.     function createProduct(address _address,string memory _nameOfProducer,
string memory _typeOfProduct,
125.     string memory _serialId, string memory _date,bytes32 _certHashM, bytes
32 _cerhHashA, bytes32 _cerhHashB) public {
126.         require (_address == manufacturer);
127.         bytes32 certHash = keccak256(abi.encodePacked(_nameOfProducer,_type
OfProduct,_serialId,_date,_certHashM,_cerhHashA,_cerhHashB,block.timestamp));
128.         productCertificates.push(certHash);
129.         productsOwnedByManufacturer[certHash] = Certificate(certHash,false)
;
130.         govtToManufacturerHistory[_certHashM].used = true;
131.         supplierAToManufacturerHistory[_cerhHashA].used = true;
132.         supplierBToManufacturerHistory[_cerhHashB].used = true;
133.         productDetails[certHash] = Product(_nameOfProducer,_typeOfProduct,_
serialId,_date,certHash,_cerhHashA,_cerhHashB,_certHashM);
134.
135.     }
136.
137.
138.     //used by user
139.     function getCertificateHistory(bytes32 _certHash) public view returns (
bytes32,bytes32,bytes32,bytes32){
140.         return(productDetails[_certHash].certificateHash_p,
141.         productDetails[_certHash].certificateHash_sup_a,
142.         productDetails[_certHash].certificateHash_sup_b,
143.         productDetails[_certHash].certificateHash_govt);
144.     }
145.
146.
147.     function getSupplierACert() public view returns (bytes32[] memory){
148.         return govtToSupplierACertificates;
149.     }
150.
151.     function getSupplierBCert() public view returns (bytes32[] memory){
152.         return govtToSupplierBCertificates;
153.     }
154.
155.     function getManufacturerCertA() public view returns (bytes32[] memory){
156.         return supplierAToManufacturerCertificates;
157.     }
158.
159.     function getManufacturerCertB() public view returns (bytes32[] memory){
160.         return supplierBToManufacturerCertificates;
161.     }
162.

```

```
163.         function getManufacturereCertG() public view returns (bytes32[] memory)
164.         {
165.             return govtToManufacturerCertificates;
166.         }
167.         function getProductCert() public view returns (bytes32[] memory){
168.             return productCertificates;
169.         }
170.
171.         function getSupplierAAddress() public view returns (address){
172.             return supplierA;
173.         }
174.
175.         function getSupplierBAddress() public view returns (address){
176.             return supplierB;
177.         }
178.
179.         function getManufacturerAddress() public view returns (address){
180.             return manufacturer;
181.         }
182.
183.         function getGovtAddress() public view returns (address){
184.             return certifier;
185.         }
186.     }
```

Listing 8: Solidity source code of the smart contract

A2: Smart contract ABI source code

```

1.  const contractAddress= 'Input contract address' ;
2.
3.  const ABI=
4.
5.  [
6.    {
7.      "constant": true,
8.      "inputs": [],
9.      "name": "getSupplierACert",
10.     "outputs": [
11.       {
12.         "name": "",
13.         "type": "bytes32[]"
14.       }
15.     ],
16.     "payable": false,
17.     "stateMutability": "view",
18.     "type": "function"
19.   },
20.   {
21.     "constant": false,
22.     "inputs": [
23.       {
24.         "name": "_address",
25.         "type": "address"
26.       },
27.       {
28.         "name": "_name",
29.         "type": "string"
30.       },
31.       {
32.         "name": "_nameOfCertificate",
33.         "type": "string"
34.       },
35.       {
36.         "name": "_date",
37.         "type": "string"
38.       },
39.       {
40.         "name": "_role",
41.         "type": "uint256"
42.       }
43.     ],
44.     "name": "createAndSendCertificate",
45.     "outputs": [],
46.     "payable": false,
47.     "stateMutability": "nonpayable",
48.     "type": "function"
49.   },
50.   {
51.     "constant": true,
52.     "inputs": [
53.       {
54.         "name": "",
55.         "type": "bytes32"
56.       }
57.     ],
58.     "name": "supplierDetails",
59.     "outputs": [
60.       {
61.         "name": "name",
62.         "type": "string"

```

```

63.         },
64.         {
65.             "name": "nameOfCertificates",
66.             "type": "string"
67.         },
68.         {
69.             "name": "date",
70.             "type": "string"
71.         },
72.         {
73.             "name": "certificateHash",
74.             "type": "bytes32"
75.         }
76.     ],
77.     "payable": false,
78.     "stateMutability": "view",
79.     "type": "function"
80. },
81. {
82.     "constant": true,
83.     "inputs": [
84.         {
85.             "name": "_certHash",
86.             "type": "bytes32"
87.         }
88.     ],
89.     "name": "getCertificateHistory",
90.     "outputs": [
91.         {
92.             "name": "",
93.             "type": "bytes32"
94.         },
95.         {
96.             "name": "",
97.             "type": "bytes32"
98.         },
99.         {
100.             "name": "",
101.             "type": "bytes32"
102.         },
103.         {
104.             "name": "",
105.             "type": "bytes32"
106.         }
107.     ],
108.     "payable": false,
109.     "stateMutability": "view",
110.     "type": "function"
111. },
112. {
113.     "constant": true,
114.     "inputs": [
115.         {
116.             "name": "",
117.             "type": "bytes32"
118.         }
119.     ],
120.     "name": "supplierBToManufacturerHistory",
121.     "outputs": [
122.         {
123.             "name": "hash",
124.             "type": "bytes32"
125.         },
126.         {
127.             "name": "used",

```



```

128.         "type": "bool"
129.     }
130. ],
131.     "payable": false,
132.     "stateMutability": "view",
133.     "type": "function"
134. },
135. {
136.     "constant": true,
137.     "inputs": [],
138.     "name": "getManufacturerCertA",
139.     "outputs": [
140.         {
141.             "name": "",
142.             "type": "bytes32[]"
143.         }
144.     ],
145.     "payable": false,
146.     "stateMutability": "view",
147.     "type": "function"
148. },
149. {
150.     "constant": true,
151.     "inputs": [
152.         {
153.             "name": "",
154.             "type": "bytes32"
155.         }
156.     ],
157.     "name": "govtToSupplierBHistory",
158.     "outputs": [
159.         {
160.             "name": "hash",
161.             "type": "bytes32"
162.         },
163.         {
164.             "name": "used",
165.             "type": "bool"
166.         }
167.     ],
168.     "payable": false,
169.     "stateMutability": "view",
170.     "type": "function"
171. },
172. {
173.     "constant": true,
174.     "inputs": [
175.         {
176.             "name": "",
177.             "type": "bytes32"
178.         }
179.     ],
180.     "name": "govtToManufacturerHistory",
181.     "outputs": [
182.         {
183.             "name": "hash",
184.             "type": "bytes32"
185.         },
186.         {
187.             "name": "used",
188.             "type": "bool"
189.         }
190.     ],
191.     "payable": false,
192.     "stateMutability": "view",

```

```

193.         "type": "function"
194.     },
195.     {
196.         "constant": true,
197.         "inputs": [
198.             {
199.                 "name": "",
200.                 "type": "bytes32"
201.             }
202.         ],
203.         "name": "govtToSupplierAHistory",
204.         "outputs": [
205.             {
206.                 "name": "hash",
207.                 "type": "bytes32"
208.             },
209.             {
210.                 "name": "used",
211.                 "type": "bool"
212.             }
213.         ],
214.         "payable": false,
215.         "stateMutability": "view",
216.         "type": "function"
217.     },
218.     {
219.         "constant": true,
220.         "inputs": [],
221.         "name": "getGovtAddress",
222.         "outputs": [
223.             {
224.                 "name": "",
225.                 "type": "address"
226.             }
227.         ],
228.         "payable": false,
229.         "stateMutability": "view",
230.         "type": "function"
231.     },
232.     {
233.         "constant": true,
234.         "inputs": [],
235.         "name": "getSupplierAAddress",
236.         "outputs": [
237.             {
238.                 "name": "",
239.                 "type": "address"
240.             }
241.         ],
242.         "payable": false,
243.         "stateMutability": "view",
244.         "type": "function"
245.     },
246.     {
247.         "constant": true,
248.         "inputs": [],
249.         "name": "getSupplierBCert",
250.         "outputs": [
251.             {
252.                 "name": "",
253.                 "type": "bytes32[]"
254.             }
255.         ],
256.         "payable": false,
257.         "stateMutability": "view",

```

```

258.     "type": "function"
259.   },
260.   {
261.     "constant": true,
262.     "inputs": [
263.       {
264.         "name": "",
265.         "type": "bytes32"
266.       }
267.     ],
268.     "name": "productsOwnedByManufacturer",
269.     "outputs": [
270.       {
271.         "name": "hash",
272.         "type": "bytes32"
273.       },
274.       {
275.         "name": "used",
276.         "type": "bool"
277.       }
278.     ],
279.     "payable": false,
280.     "stateMutability": "view",
281.     "type": "function"
282.   },
283.   {
284.     "constant": true,
285.     "inputs": [
286.       {
287.         "name": "",
288.         "type": "bytes32"
289.       }
290.     ],
291.     "name": "manufacturerDetails",
292.     "outputs": [
293.       {
294.         "name": "name",
295.         "type": "string"
296.       },
297.       {
298.         "name": "nameOfCertificates",
299.         "type": "string"
300.       },
301.       {
302.         "name": "date",
303.         "type": "string"
304.       },
305.       {
306.         "name": "certificateHash",
307.         "type": "bytes32"
308.       }
309.     ],
310.     "payable": false,
311.     "stateMutability": "view",
312.     "type": "function"
313.   },
314.   {
315.     "constant": true,
316.     "inputs": [],
317.     "name": "getProductCert",
318.     "outputs": [
319.       {
320.         "name": "",
321.         "type": "bytes32[]"
322.       }

```

```

323.         ],
324.         "payable": false,
325.         "stateMutability": "view",
326.         "type": "function"
327.     },
328.     {
329.         "constant": true,
330.         "inputs": [
331.             {
332.                 "name": "",
333.                 "type": "bytes32"
334.             }
335.         ],
336.         "name": "supplierAToManufacturerHistory",
337.         "outputs": [
338.             {
339.                 "name": "hash",
340.                 "type": "bytes32"
341.             },
342.             {
343.                 "name": "used",
344.                 "type": "bool"
345.             }
346.         ],
347.         "payable": false,
348.         "stateMutability": "view",
349.         "type": "function"
350.     },
351.     {
352.         "constant": false,
353.         "inputs": [
354.             {
355.                 "name": "_address",
356.                 "type": "address"
357.             },
358.             {
359.                 "name": "_nameOfProducer",
360.                 "type": "string"
361.             },
362.             {
363.                 "name": "_typeOfProduct",
364.                 "type": "string"
365.             },
366.             {
367.                 "name": "_serialId",
368.                 "type": "string"
369.             },
370.             {
371.                 "name": "_date",
372.                 "type": "string"
373.             },
374.             {
375.                 "name": "_certHash",
376.                 "type": "bytes32"
377.             }
378.         ],
379.         "name": "createCertificateForManufacturer",
380.         "outputs": [],
381.         "payable": false,
382.         "stateMutability": "nonpayable",
383.         "type": "function"
384.     },
385.     {
386.         "constant": true,
387.         "inputs": [],

```

```

388.         "name": "getManufacturerAddress",
389.         "outputs": [
390.             {
391.                 "name": "",
392.                 "type": "address"
393.             }
394.         ],
395.         "payable": false,
396.         "stateMutability": "view",
397.         "type": "function"
398.     },
399.     {
400.         "constant": true,
401.         "inputs": [],
402.         "name": "getSupplierBAddress",
403.         "outputs": [
404.             {
405.                 "name": "",
406.                 "type": "address"
407.             }
408.         ],
409.         "payable": false,
410.         "stateMutability": "view",
411.         "type": "function"
412.     },
413.     {
414.         "constant": true,
415.         "inputs": [
416.             {
417.                 "name": "",
418.                 "type": "bytes32"
419.             }
420.         ],
421.         "name": "productDetails",
422.         "outputs": [
423.             {
424.                 "name": "nameOfProducer",
425.                 "type": "string"
426.             },
427.             {
428.                 "name": "typeOfProduct",
429.                 "type": "string"
430.             },
431.             {
432.                 "name": "serialId",
433.                 "type": "string"
434.             },
435.             {
436.                 "name": "date",
437.                 "type": "string"
438.             },
439.             {
440.                 "name": "certificateHash_p",
441.                 "type": "bytes32"
442.             },
443.             {
444.                 "name": "certificateHash_sup_a",
445.                 "type": "bytes32"
446.             },
447.             {
448.                 "name": "certificateHash_sup_b",
449.                 "type": "bytes32"
450.             },
451.             {
452.                 "name": "certificateHash_govt",

```

```

453.         "type": "bytes32"
454.     }
455. ],
456.     "payable": false,
457.     "stateMutability": "view",
458.     "type": "function"
459. },
460. {
461.     "constant": true,
462.     "inputs": [],
463.     "name": "getManufacturereCertG",
464.     "outputs": [
465.         {
466.             "name": "",
467.             "type": "bytes32[]"
468.         }
469.     ],
470.     "payable": false,
471.     "stateMutability": "view",
472.     "type": "function"
473. },
474. {
475.     "constant": true,
476.     "inputs": [],
477.     "name": "getManufacturerCertB",
478.     "outputs": [
479.         {
480.             "name": "",
481.             "type": "bytes32[]"
482.         }
483.     ],
484.     "payable": false,
485.     "stateMutability": "view",
486.     "type": "function"
487. },
488. {
489.     "constant": false,
490.     "inputs": [
491.         {
492.             "name": "_address",
493.             "type": "address"
494.         },
495.         {
496.             "name": "_certHash",
497.             "type": "bytes32"
498.         }
499.     ],
500.     "name": "sendCertificateToManufacturer",
501.     "outputs": [],
502.     "payable": false,
503.     "stateMutability": "nonpayable",
504.     "type": "function"
505. },
506. {
507.     "constant": false,
508.     "inputs": [
509.         {
510.             "name": "_address",
511.             "type": "address"
512.         },
513.         {
514.             "name": "_nameOfProducer",
515.             "type": "string"
516.         }
517.     ],

```

```

518.         "name": "_typeOfProduct",
519.         "type": "string"
520.     },
521.     {
522.         "name": "_serialId",
523.         "type": "string"
524.     },
525.     {
526.         "name": "_date",
527.         "type": "string"
528.     },
529.     {
530.         "name": "_certHashM",
531.         "type": "bytes32"
532.     },
533.     {
534.         "name": "_cerhHashA",
535.         "type": "bytes32"
536.     },
537.     {
538.         "name": "_cerhHashB",
539.         "type": "bytes32"
540.     }
541. ],
542. "name": "createProduct",
543. "outputs": [],
544. "payable": false,
545. "stateMutability": "nonpayable",
546. "type": "function"
547. },
548. {
549.     "inputs": [
550.         {
551.             "name": "_addressA",
552.             "type": "address"
553.         },
554.         {
555.             "name": "_addressB",
556.             "type": "address"
557.         },
558.         {
559.             "name": "_addressM",
560.             "type": "address"
561.         },
562.         {
563.             "name": "_government",
564.             "type": "address"
565.         }
566.     ],
567.     "payable": false,
568.     "stateMutability": "nonpayable",
569.     "type": "constructor"
570. }
571. ]
572. export {
573.     contractAddress,
574.     ABI,
575. }

```

Listing 9: Smart contract ABI source code

Appendix B:

Graphical user interfaces

B1: User interface source code

The source code of the user interface collects all important information about all created virtual identities in order to display them for the users. All other interfaces use a structure which is based on the source code shown below (Listing 10). Therefore, all other interfaces are only illustrated in their final graphic layout.

```
1. import React, { Component } from 'react';
2. import {
3.   Card,
4.   CardBody,
5.   Table,
6.   Button, Modal, ModalHeader, ModalBody, ModalFooter, FormGroup, Label, Input,
7. } from 'reactstrap';
8. import { contractAddress, ABI } from './Constants';
9.
10. const isEthereumAddress = require('is-ethereum-address');
11. const Web3 = require('web3');
12.
13.
14. export default class Certificate extends Component {
15.
16.   constructor(props) {
17.     super(props);
18.     this.state = {
19.       modal: false,
20.       orgName: '',
21.       certName: '',
22.       date: '',
23.       role: '0',
24.       pubKey: '',
25.       isKey: false,
26.       selectedHash: '',
27.       getProductArr: [],
28.       productsArr: [],
29.       pbKey:String
30.
31.
32.     };
33.
34.     this.toggle = this.toggle.bind(this);
35.   }
36.
37.   componentDidMount() {
38.     this.getAllProducts();
39.   }
40.
41.
42.   async getAllProducts() {
43.
44.
45.     var web3;
```



```

46. let ethereum = window.ethereum;
47. console.log('ethereum', window.ethereum);
48.
49. web3 = new Web3(window.web3.currentProvider);
50.
51. const account = await ethereum.enable();
52. const MyContract = new web3.eth.Contract(ABI, contractAddress);
53.
54. console.log(MyContract.methods);
55. const accounts = await web3.eth.getAccounts();
56.
57. try {
58.   MyContract.methods.getProductCert().call({ from: accounts[0] })
59.     .then((result) => {
60.       console.log(result);
61.       this.setState({ getProductArr: [...result] });
62.
63.
64.     });
65.
66.   MyContract.methods.getManufacturerAddress().call({from:accounts[0]})
67.     .then((res)=>{
68.       console.log(res)
69.       this.setState({pbKey:res})
70.     })
71.   } catch (e) {
72.     console.log(e.message);
73.   }
74.
75. }
76.
77.
78. async getProductDetails(e) {
79.
80.
81.   var web3;
82.   let ethereum = window.ethereum;
83.   console.log('ethereum', window.ethereum);
84.
85.   web3 = new Web3(window.web3.currentProvider);
86.
87.   const account = await ethereum.enable();
88.   const MyContract = new web3.eth.Contract(ABI, contractAddress);
89.
90.   console.log(MyContract.methods);
91.   const accounts = await web3.eth.getAccounts();
92.
93.   try {
94.     MyContract.methods.getCertificateHistory(e).call({ from: accounts[0] })
95.       .then((result) => {
96.         console.log(result['0']);
97.         for (var key in result) {
98.           this.state.productsArr.push(result[key]);
99.         }
100.         this.setState({ productsArr: this.state.productsArr });
101.         console.log('Details Obj', this.state.productsArr);
102.         // console.log(this.state.getProductDetails['0'])
103.
104.
105.       });
106.     } catch (e) {
107.       console.log(e.message);
108.     }
109.
110.   }

```

```

111.
112.     selectRow(e) {
113.         console.log('Target Value', typeof (e), e);
114.
115.
116.         this.getProductDetails(e);
117.         this.setState({ pubKey: e, modal: true });
118.
119.
120.     }
121.
122.     toggle() {
123.
124.         this.setState(prevState => ({
125.             modal: !prevState.modal,
126.             productsArr: [],
127.
128.         }));
129.     }
130.
131.     render() {
132.         return (
133.             <div>
134.                 <Card>
135.                     <CardBody style={{ display: 'flex', alignItems: 'flex-
136.                         end', justifyContent: 'flex-end' }}>
137.                         /* <Button color="primary" onClick={this.toggle} >Send Part A
138.                         </Button> */
139.                         <Modal isOpen={this.state.modal} toggle={this.toggle} style={{
140.                         marginTop: '180px' }}>
141.                             <ModalHeader toggle={this.toggle}>Hash ID Information
142.                             </ModalHeader>
143.                             <ModalBody>
144.                                 <FormGroup>
145.                                     <Label for="name">Address of producer</Label>
146.                                     <Input type="text" name="name" id="name" value={this.stat
147.                                     e.pbKey}
148.                                     placeholder="Enter public key here ... "/>
149.
150.                                     <Label for="name">Product - Hash ID</Label>
151.                                     <Input type="text" name="name" id="name" value={this.stat
152.                                     e.productsArr[0]}
153.                                     placeholder="Enter public key here ... "/>
154.
155.                                     <Label for="name">Rear Fork - Hash ID</Label>
156.                                     <Input type="text" name="name" id="name" value={this.stat
157.                                     e.productsArr[1]}
158.                                     placeholder="Enter public key here ... "/>
159.
160.                                     <Label for="name">Footboard - Hash ID</Label>
161.                                     <Input type="text" name="name" id="name" value={this.stat
162.                                     e.productsArr[2]}
163.                                     placeholder="Enter public key here ... "/>
164.
165.                                     <Label for="name">Certificate - Hash ID</Label>
166.                                     <Input type="text" name="name" id="name" value={this.stat
167.                                     e.productsArr[3]}
168.                                     placeholder="Enter public key here ... "/>
169.
170.                                 </FormGroup>
171.                             </ModalBody>
172.                             <ModalFooter>
173.                                 <Button color="primary" onClick={this.toggle}>Send</Button>
174.                             </ModalFooter>
175.                         </Modal>
176.                     </CardBody>
177.                 </Card>
178.             </div>
179.         );
180.     }

```

```

165.         <Button color="secondary" onClick={this.toggle}>Cancel</But
ton>
166.             </ModalFooter>
167.         </Modal>
168.     </CardBody>
169. </Card>
170.
171.
172.     <Card>
173.         <CardBody>
174.
175.             <Table hover>
176.                 <thead>
177.                     <tr>
178.                         <th style={{ fontWeight: 'bold' }}>#</th>
179.                         <th style={{ fontWeight: 'bold' }}>Hash ID</th>
180.
181.                     </tr>
182.                 </thead>
183.                 <tbody>
184.                     {
185.                         this.state.getProductArr.map((val, inx) => {
186.                             return (
187.
188.
189.                                 <tr>
190.                                     <td>{inx}</td>
191.                                     <td onClick={() => this.selectRow(val)}>{val} </td>
192.                                 </tr>
193.                             );
194.
195.                         })
196.                     }
197.
198.                 </tbody>
199.             </Table>
200.         </CardBody>
201.     </Card>
202.
203. </div>
204. );
205. }
206. }

```

Listing 10: React user interface source code

B2: User interface layout

Currently owned by: ×
0x61940bdc9daf18b0F83F9d30F527416f375F1A6C

Address of producer
0x61940bdc9daf18b0F83F9d30F527416f375F1A6C

Assembly – Hash ID
0x3a7982c774df10328783e73f2ab6a7c62325c1cf80f6f

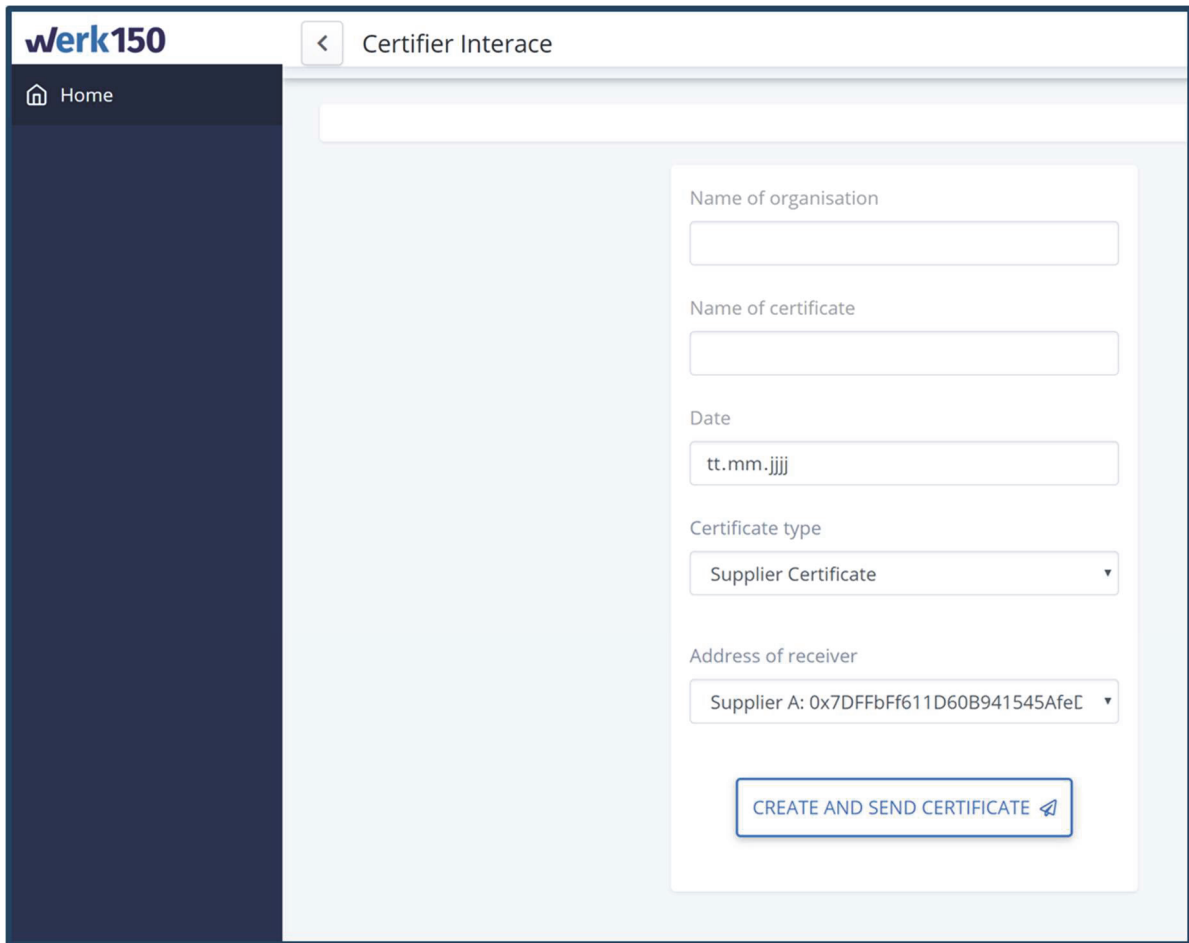
Footboard – Hash ID
0xcad3fed2ebf1b8de2ce86b9df0ce90ecc4b9004750cck

Rear fork – Hash ID
0xd647d48eacacb4a7eb68e621e5457bb0cfbf267fb35d

Certificate – Hash ID
0x18bf7f13b213b70467878d7f9828b507d1e8e958efa5

Figure 53: User interface layout

B3: Certifier interface layout



The screenshot shows the 'Certifier Interface' in the 'werk150' system. The interface includes a dark blue sidebar with a 'Home' link and a main content area with a light blue background. The main content area contains a form with the following fields:

- Name of organisation:
- Name of certificate:
- Date:
- Certificate type: ▾
- Address of receiver: ▾

At the bottom of the form is a button labeled 'CREATE AND SEND CERTIFICATE' with a right-pointing arrow icon.

Figure 54: Certifier interface layout

B4: Supplier interface layout

The screenshot shows the 'Supplier Interface' page in the Werk150 system. On the left is a dark blue sidebar with 'Home' and 'Parts' options. The main content area contains a form with the following fields and buttons:

- Name of organisation:** A text input field.
- Type of part:** A text input field.
- ERP-System ID:** A text input field.
- Date:** A text input field with the placeholder 'tt.mm.jjjj'.
- SELECT CERTIFICATE:** A solid blue button.
- CREATE PART:** A button with a white background and a blue border, featuring a right-pointing arrow icon.

Figure 55: Supplier interface layout

B5: Manufacturer interface layout

The screenshot displays the 'Manufacturer Interface' within the Werk150 system. On the left, a dark blue sidebar contains navigation options: 'Home' (with a house icon) and 'Products' (with a document icon). The main content area is light blue and features a white form titled 'Manufacturer Interface' with a back arrow icon. The form contains the following fields and buttons:

- Name of organisation:** A text input field.
- Type of product:** A text input field.
- ERP-System ID:** A text input field.
- Date:** A text input field with a placeholder 'tt.mm.jjjj'.
- Select Certificate:** A blue button.
- Select Footboard:** A blue button.
- Select Rear fork:** A blue button.
- CREATE PRODUCT:** A white button with a blue border and a right-pointing arrow icon.

Figure 56: Manufacturer interface layout