FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Component provenance tracing through blockchain-based, trackable data exchange for safety critical industrial supply chains

Sangeeta Joseph

FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Component provenance tracing through blockchain-based, trackable data exchange for safety critical industrial supply chains

Rückverfolgung der Komponenten Herkunft in sicherheitskritischen industriellen Lieferketten durch einen blockchainbasierten, trackbaren Datenaustausch

Author:	Sangeeta Joseph
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	M.Sc. Dian Balta (fortiss)
	M.Sc. Ulrich Gallersdörfer
Submission Date:	November 15, 2020

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, November, 15, 2020 Sangeeta Joseph

Acknowledgements

I would like to begin by thanking Prof. Dr. Florian Matthes for allowing me to engage in such an exciting topic for this thesis at his chair for Software Engineering for Business Information Systems, for which I am deeply grateful.

I also want to thank my advisors Dian Balta and Ulrich Gallersdörfer, for helping me find this topic and quickstart my work. I then want to express my gratitude to Anastasios Kalogeropoulos and Matthias Buchinger for having been present during every step of this development to support my queries and push me in the right direction.

Finally, I thank my boyfriend Maik and my friends for their emotional support and infinite patience throughout this journey. My endless appreciation goes to my family, who has made this opportunity a reality for me through their sacrifice and unrelenting support for my choices.

Abstract

Traditional business relations between members in a supply chain context involve lengthy and error prone processes of communication with little reliability and safety, which result in low trust environments where product and document tampering is hard to avoid. This is a fatal condition for high risk manufacture industries where product history must be reliable beyond doubt to ensure the end result is to be trusted. In this dissertation we introduce the concepts of blockchain as the best tamper proof network for transactions, we show how such technology could benefit private supply chain scenarios by delivering a transparent data flow from raw material supplier all the way to the end customer while guaranteeing efficient traceability, trust among peers and privacy.

Contents

List of Figures

List of Tables

List of Abbreviations

SCM	Supply Chain Management
SME	Small Medium-Sized Enterprise
IPFS	InterPlanetary File System
SME	Small Medium-Sized Enterprise
IOT	Internet of Things
DSRM	Design Science Research Methodology
ERP	Enterprise Resource Planning
CAD	Computer-aided Design
GDPR	General Data Protection Regulation
BPMN	Business Process Model and Notation
SDK	Software Development Kit
UML	Unified Modeling Language
DOM	Document Object Model
UI	User Interface
API	Application Programming Interface

1 Introduction

1.1 Motivation

A supply chain is a network of organizations that share the goal of creating and distributing high-quality products or services [1]. Before the First Industrial Revolution, the supply chain was reasonably straightforward, where resources were extracted and transported to skilled artisans who manufactured finished products later sold in the markets [2]. Today's supply chains are much more complicated, where the system consists of not only manufacturers and suppliers but also assemblers, warehouse managers, logistics companies for transportation, distributors, retailers, and also consumers [3]. Often, the stakeholders involved have no knowledge of other participants in the supply chain [4], especially the consumer who might not be aware of where, when, or under which conditions the products have been manufactured.

As defined by the Council of Supply chain management professionals, "Supply Chain Management (SCM) encompasses the planning and management of all activities involved in sourcing and procurement, conversion, and all logistics management activities. Importantly, it also includes coordination and collaboration with channel partners, which can be suppliers, intermediaries, third-party service providers, and customers." [**31**]. In short, supply chain management involves the management of products, information, and funds.

Because of globalization, SCM is now prone to risks due to the various interconnected parties, and a minor fault could result in negative consequences for all members [5]. Risks include theft, counterfeit, raw material price fluctuations, natural disaster, logistic delays, environmental hazards, economic instability, and supplier inconsistency [6]. Out of these, counterfeit and fraud account for 11% of the total risks [6]. A study conducted found that around 33% of 1215 fish samples collected were mislabeled [7].

An example of a company relying on the supply chain is IBO. IBO GmbH is a Small Medium-Sized Enterprise (SME) manufacturing roller bearings mostly used in safety-critical domains such as aerospace, robotics, medical and transportation. Because of this, the products have to be thoroughly tested and certified following strict safety standards.

A counterfeit sold as an IBO product is a massive risk to the company's liability, and, at present, it's difficult to prove the authenticity of their product. The consequences of using counterfeit products could result in a considerable reputation, financial, material, and, in the worst case, life loss. This could be minimized by delivering a Traceable Product History without disclosing business secrets. A Traceable Product History is the chronological sequence of steps taken in a supply chain. The steps are actions taken by a member in the supply chain that change the state of the product, including submitting an order, acquiring a raw natural resource (materializing as a product), testing, altering a component, combining two or more products into a single one, transfer in ownership or responsibility. This product becomes now an asset and each step is to be considered as a transaction on this asset with data (information) associated with it.

Transactions and relevant information are traditionally stored in volatile documents that can be lost, un/intentionally omitted, or tampered with and need to be continuously duplicated and manually synced between parties. A client's inquiry on a product can take weeks to be consolidated because data needs to be gathered from multiple sources and parties via the mentioned inefficient data flow. Data flow occurs during a transfer of ownership or change of status between the two parties. In worse cases, disputes between parties can lead to third parties' involvement to resolve a claim.

These problems can be solved by a transparent data flow, which can be achieved by having a tamper-proof single data source such as a distributed ledger technology, a blockchain-based system that keeps track of records, product information, and history. Such system treats the asset and transactions as digital ones. A client accessing data from a blockchain can easily verify a product's authenticity and detect counterfeit products. Depending on the blockchain used, systems can be implemented to provide different access levels for their users. In addition to transparency, the blockchain-based system offers accountability and scalability through multiple organizations and locations. Adopting blockchain could help the supply chain industry reduce risks, improve management, and increase consumer and manufacturer trust.

1.2 Purpose and Research Questions

The goal of this master's thesis is to design and implement a blockchain-based supply chain system for IBO GmbH that provides transparent data flow, traceable product history without disclosing business secrets. Based on the Design Science Research methodology [8], the focus of this thesis is summarized in the following three research questions:

1. **What are the requirements for a blockchain-based supply chain system to reduce fraud and improve supply chain management?**
   The goal is to analyze IBO GmbH's needs and desires and agree on a set of requirements.
   Methodology Followed: The first step is gathering requirements from the necessary stakeholders by conducting a meeting or interviews, examining the existing system, and studying the supply chain documentation. The next step is analyzing these requirements and modeling them using diagrams and flowcharts.
   Expected Results: A Requirement Specification Document that serves as a base for the future architecture design and implementation phase.

2. **What is the architecture of a blockchain-based system for fraud reduction and supply chain management?**
   Based on the requirements specified, the architecture of a system needs to be designed and evaluated.
   Methodology Followed: Examine existing blockchain systems for supply chain, research on the available frameworks and tools, design the components of the system based on the requirements, define process level diagrams to explain the system, and eventually perform an architectural evaluation.
   Expected Results: A reference architecture using the 4+1 view model architecture [9].

3. **What is the prototypical implementation for a Blockchain-Based Supply Chain system for fraud reduction and supply chain management?**
   Methodology Followed: Implementing the various architectural components designed in the previous research question and the evaluation of the system concerning the requirements.
   Expected Results: A functionally working prototype that helps IBO and other SMEs to detect any fraud and minimize the time spend during audits.

1.3 Outline

Chapter 2 introduces state of the art blockchain technology and relevant comparisons followed by descriptions of related blockchain-based system. Chapter 3 focuses on introducing the Design Science Research Methodology and describing the research approach followed. Chapter 4 presents the requirements analysis consisting of the use-case diagrams, system analysis, functional and non-functional requirements established for the thesis. In Chapter 5, the system design and architecture are presented using 4+1 view model. Furthermore, the evaluation of the requirements concerning architecture and prototypical implementation is reported in Chapter 6. The thesis is concluded in Chapter 7, covering the limitations and the future works in this field.

2 Theoretical Background and Related work

This chapter provides description of the state of the art in blockchain technology and a comparison of the relevant systems. Additionally, details about the existing system using blockchains in supply chains are also discussed.

2.1 Blockchain Technology

In financial terms, a ledger is a record-keeping system containing all business transactions [10]. Similarly, a distributed ledger is a unique database controlled by a set of participants in a distributed environment [11]. The data can be accessed or updated across different sites or geographic locations and by multiple people. The network participants own an identical copy of the ledger, and any update to it is reflected in all copies, making the ledger safe from external attacks.

Although used as synonyms, the blockchain is a type of distributed ledger technology. A blockchain is an immutable peer-to-peer ledger consisting of ordered units known as cryptographically secure blocks. The block consists of a timestamp, set of transactions, and the hash of the previous block. A transaction is an order that updates the current ledger and is validated by a consensus mechanism and grouped with other transactions into blocks appended to every participant's ledger. Given the hash of the last block, it's possible to traverse all the way to the first block in the ledger.

Blockchain inherently provides advantages over centralized systems. Some of the features include:

1. **Transparency:** As each node on the blockchain network owns a copy of the ledger, all the participants have access to the transactions and data, and hence the system is transparent.
2. **Immutability:** Once the transactions are added to the blockchain, it is nearly impossible to change it, making the ledger immutable.

3. **Decentralized:** The blockchain is not governed by a single authority but it's maintained by the nodes in the network.
4. **Increased Security:** With the blockchain being decentralized, there is no single point of failure. A copy of the ledger is stored on each node, and the consensus mechanism is used to ensure each copy's integrity.

2.1.1 Public and Private blockchain

Blockchain is classified into three major types: public, private, and hybrid [32] and Table 2.1 describes the difference between them.

1. **Public Blockchain:** Allows any participant to join the network as the blockchain has no single central governance. All nodes in the network have the authority to receive a copy of the ledger and submit transactions to the blockchain [12][13]. It is nearly impossible to corrupt the ledger and it's extremely secure. Bitcoin and Ethereum fall into this category[14].
2. **Private Blockchains:** They are also referred to as permissioned blockchains. The network can be partially or entirely centralized, i.e., the owner of the network is responsible for providing access to participants to view and update the ledger [12][13]. Private blockchains develop solutions for enterprise systems where entities need to be verified before entering the network. Examples of this include Hyperledger Fabric and Hyperledger Sawtooth [14].
3. **Hybrid Blockchains:** Also introduced as Consortium blockchains, it is a combination of public and private blockchains. The chain is governed by a group of equally powerful parties [14]. The network is partly decentralized because the copies of the ledger are distributed only to authorized entities [12]. Hybrid blockchains could help cases where various organizations in a field who don't trust each other want to share information by carrying out transactions [15].

2.1.2 Bitcoin

One of the very first recognized implementations of blockchain technology is Bitcoin. Launched in 2009, Bitcoin was introduced as a digital currency that would allow online payments from one party to another without a central authority like banks or the government (for this reason called cryptocurrency) [16]. Unlike fiat currency, bitcoins don't have a physical existence; they are created, stored, and traded with blockchain

	Public	Private	Hybrid
Requires Permission?	No	No	Yes
Central Authority?	No	Single Organization	Multiple Organizations
Decentralized	Yes	Partial or Completely Centralized	Partially
Performance	Slow	Fast	Fast
Scalability	Low	High	High
Energy Consumptions	High	Low	Low
Transaction Data Privacy	None	Controlled	Controlled
User Anonymity	Yes	No	No

Table 2.1: Difference between public, private and hybrid blockchains

technology. Bitcoin is quite popular as a cryptocurrency paved the way for launching many other cryptocurrencies called Altcoins.

The Bitcoin network consists of nodes called miners responsible for curing the network by processing transactions by solving complex math problems using high powered computers. Miners are rewarded for adding a block to the blockchain and the form transaction fees is paid in bitcoin. Since the success of Bitcoin, several applications have tried to leverage the benefits of distributed ledger technologies.

2.1.3 Ethereum

Ethereum, another cryptocurrency, introduced a new feature not present in Bitcoin. By introducing the concept of smart contracts, Ethereum stood out as a platform to develop decentralized applications [17]. According to Szabo [18], "a smart contract is a set of promises, specified in digital form, including protocols within which the parties perform on these promises.” In simple terms, Smart contracts are computer programs running on Ethereum network that contain business logic [19].

2.1.4 Corda

Corda is a distributed ledger technology developed by the R3 Consortium [20]. The consortium consists of more than 100 well-known banks that focused on developing solutions for the financial sector. Data privacy is one of the main features address in the Corda blockchain, i.e., data is shared amongst relevant parties [20]. CorDapps are applications developed on the Corda and use a virtual machine, i.e., Java Virtual Machine, and the contracts are written in Java [21].

2.2 Hyperledger Fabric

Hyperledger fabric is an enterprise-grade, open-source, Distributed Ledger Technology (DLT) platform used to develop enterprise applications leveraging the features of distributed ledgers and blockchain technology. Fabric is a permissioned blockchain [22] with a modular and configurable architecture, allowing developers to adopt different components depending on the use-case.

The core components of the Hyperledger Fabric protocol are as follows [23]:

- the assets, which are the element of exchange and can be monetary, physical/digital property, or anything defined by the participants
- the distributed ledger, which decentralizes the network data by storing a copy of the current state and transaction history in each member of the blockchain
- the smart contract (aka chaincode), which acts as a strict protocol to gate the access and addition of data in the blockchain
- the consensus, which is the mechanism by which the participating members approve transactions on the blockchain

2.2.1 Architecture

The hyperledger blockchain is the infrastructure of the ledger and chaincode. Usually, a blockchain network is initiated by an administrator organization that issues the network policies (Network Configuration) and the Ordering Service [23].

An **Ordering service** is used to maintain the order of the transactions in the blocks. There can be multiple Ordering Services associated with different Network Configuration, depending on the network structure [24].

A **certificate authority** is issued for the network, which dispenses certificates used to identify components belonging to the organizations [25]. The mapping between the certificates to the member organizations is achieved via the Membership Service Provider, accessed by the Network Configuration [25]. The Network Configuration can be updated to add and remove administrators to the network, which will share the administrative rights to the associated Ordering Service.

When new organizations are added to the network, each has its own Certificate Authority, and the administrators can group two or more organizations into a **consortium**, which is the set of organizations that need to perform a transaction among each other. Members of a consortium communicate via **Channels**, which are private mechanisms to exchange data. Channels have their own configuration called Channel Configuration, with their own policies, independent of the Network Configuration, and only members of the consortium have power over the configuration [23]. A channel is attached to an Ordering Service. Each channel hosts the ledger, which is physically stored into a Peer Node.

A **Client Application**, external to the network, communicates with the network via the channel by following the Channel Configuration rules and having a valid identity. The application cannot directly access the ledger of the channel, and the only way it can query or update it is by using the chaincode [23]. A Chaincode definition must always be approved by most consortium members before being associated with the channel and installed on the peers. Part of the chaincode definition is the Endorsement policy, which describes which organizations must approve transactions. Once the chaincode is installed on the peer, a client application can be invoked by sending a transaction proposal, which triggers the endorsement policy and responds to the client application.

Peer nodes are the essential building blocks of the blockchain. A peer contains one or more instances of a ledger and has installed one or more smart contracts [26]. Client applications interact with peers to perform a query or update of the ledger. The connection to a peer, the invocation of a chaincode, and the proposal submission are performed via an API call. A query request sent to the peer results in an invocation of the ledger's local instance, and the result immediately returned to the application. In general, there is no need for the peer node to interact with other nodes for query requests unless the local instance of the ledger is out of date. In case of an update transaction, the peer would invoke only its ledger's local instance; however, it would invoke all other peers in the network for the consensus process. This returns a proposed update from the consensus of all nodes, which is returned to the application. At the same time, the consensus invokes the Orderer to packages the transactions into a block, which then is distributed to all peers for them to perform a local update to the ledger. At the end of this asynchronous operation, the application is notified (asynchronously).

Assets in the ledger are represented as key/value pairs, where the value is a binary or JSON description and are modifiable only through chaincode transactions [23]. The chaincode defines the rules that the actors involved must obey to update the key/ledger's value pairs, effectively defining how the business is conducted towards the asset. The transaction history or sequence of tamper-resistant records of all state changes performed via chaincode is stored in the blockchain, combined with the ledger's assets' current state. The scope of a ledger is limited to a channel that can be public (visible to all participants in the network) or private between a subset of actors.

Data stored in the ledger can be accessed by all the peers in the channel. To exchange private data on the channel, one can create a new channel and add the organizations that need to access the data. However, adding a new channel for each use case is inefficient; hence hyperledger offers the ability to create **private data collections** [27]. The collection consists of two elements, the private data and the hash to the private data. The private data hash is then uploaded on the ledger, which can be accessed by all the parties in the channel to validate its state [27].

2.2.2 Steps to setup a basic hyperledger network

This section explains how the organizations collaborate with different components and describes the steps to setup up a basic hyperledger network. Figure 2.1 illustrates the basic hyperledger fabric network referred from the official documentation [28].

1. Create a Network: A hyperledger network N is created when the orderer is up and running. Organization R4 is the initial admin who sets up the network by adding orderer O4, configured using the network configuration NC4. Certificate Authority CA4 is responsible for dispensing identities to nodes of the C4 organization.
2. Add R1 as a network admin: Update the network configuration NC4 to make R1 the admin. Orderer O4 runs on organization R4's infrastructure, and R1 shares the admin rights over it.

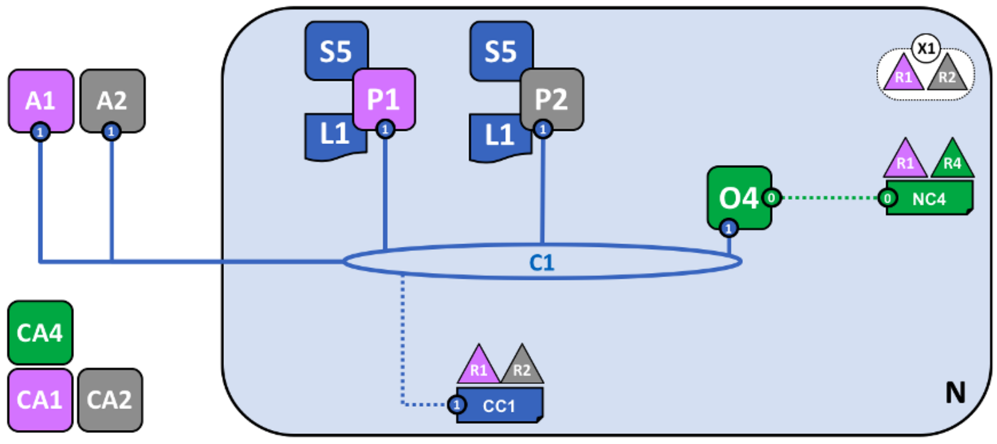


Figure 2.1: Hyperledger Fabric network [28]

3. Define a Consortium: A consortium is defined as a group with a common goal. The network admin creates a consortium X1 and adds organization R1 and R2 with respective certificate authorities CA1 and CA2 to it.
4. Create a Channel for the Consortium: Channel C1 is created for the consortium X1 and is configured by the channel configuration CC1 managed by R1 and R2.
5. Add Peers and Ledgers: Peer P1 of Organization R1 joins the channel C1. P1 hosts the ledger L1 and can communicate with the orderer O4 through the channel.
6. Install Chaincode and Client Application: Smart contract chaincode S5 is installed on the peer P1. And client application A1 of organization R1 updates the ledger L1 with the help of the smart contracts.
7. Adding new Peers: Similarly, peer P2 joins the channel C1 and holds the copy of the ledger L1 and deployed smart contract S5. Client application A1 and A2 can access the ledger L1 by invoking the intelligent contracts on either of the peers P1 or P2.

2.3 Comparison between Ethereum, Corda and Hyperledger Fabric

This section provides a brief analysis of the different characteristics of Ethereum, Corda, and Hyperledger fabric that helps determine a DLT technology for implementing the prototype. Ethereum was created to provide a platform to build distributed applications; on the other hand, Corda's use cases focused on solutions only for financial industries, while the modular nature of hyperledger provides solutions to multiple industries.

Ethereum is a public-based blockchain and, therefore, not suitable to store confidential data, while Corda and Hyperledger can be used as a permissioned or a private blockchain and have mechanisms to store private data.

In Ethereum, all parties in the network need to reach a consensus on the transactions' order to keep the ledger's state consistent. This is achieved by mining based on the Proof-of-work consensus scheme. In contrast, Corda and hyperledger have subsets of peers involved in the transaction process and the consensus algorithm is chosen based on the application's requirements. This results in Ethereum having a low-performance compared to hyperledger and Corda.

Smart contracts are just business logic written using a programming language. The smart contracts can be implemented using Solidity for Ethereum, Go, and Java for Hyperledger Fabric and Kotlin and Java for Corda.

Tokenization of assets can be quickly done in Ethereum using the ERC20 token standards, and similarly can be done in Corda using its token sdk. Hyperledger, on the other hand, needs to implement custom solutions to tokenize an asset. Table 2.2 summarizes the differences between these blockchain technologies.

2.4 IPFS

InterPlanetary File System (IPFS) is a distributed peer-to-peer file system for blockchain-based content [29]. The content is split into different parts and stored on multiple nodes, and can be accessed using the hash of the content.

In traditional location-based addressing, such as HTTP, data can be addressed using the server's hostname. On the other hand, IPFS is based on content-addressing which utilizes the content to address the data from the network [29]. By using the hash of

	Ethereum	Corda	Hyperledger Fabric
Purpose	Platform to build distributed apps	Solutions for Financial Industries	Modular platform to build enterprise solutions
Permissions?	Public blockchain	Permissioned blockchain	Permissioned blockchain
Consensus	Mining based on proof-of-work	Parties involved in the transaction make the decision	Pluggable consensus mechanism
Transaction rate and Performance	Slow	Fast	Fast
Smart contract language	Solidity	Java and Kotlin	Go, Node.js and Java
Tokenization	Using ERC20 token standard	Using token sdk	Custom implementation

Table 2.2: Difference between Ethereum, Corda and Hyperledger Fabric

the data, the user can access the content, and the network will always return the same content irrespective of who uploads it [29].

Limitations of IPFS

Data stored on the network is not persistent, i.e., the nodes in the network decide if it is convenient to keep the content [30]. To ensure the storage of content, the user needs to pin the files to the node to avoid garbage collection. Pinning services run multiple nodes that allow the user to pin their file for a small fee. Examples of pinning services are Infura, pinata, and Temporal [30].

Another challenge concerning IPFS is privacy related. Files in IPFS can be accessed if one possess the hash of the content, hence uploading sensitive files on the network is not the right approach. One of the solutions involves encrypting the file with the recipient's public key, and the recipient can access the file and decrypt it using their private key. With this approach, only the recipient can access the file on the network.

2.5 Related projects

2.5.1 Everledger

Everledger is an independent technology company aiming to provide blockchain-based solutions in luxury goods, wine, and diamonds. These industries are facing challenges related to authenticity and sustainability. Using blockchain technology, they aim to provide solutions that help their clients verify a product's authenticity. For the diamond industry, Everledger uses blockchain, AI, nanotechnology, and Internet of Things (IOT) to create a digital record of each diamond to provide transparency for ethical sourcing [31].

This digital record represents the final product (the diamond) from beginning to end of the supply chain, which is useful for luxury brands aiming at fighting counterfeit products, but results in a traceable history composed of just ownership transfers. For a company like IBEO, a ball bearing is a made of multiple components and raw materials from several different sources, which are processed and combined, potentially by multiple manufacturers, so it cannot be limited to ownership transfer. Moreover, the final product comes into existence only at a certain step of the chain, while the predecessor products (raw material or intermediate component) cease to exist. This kind of digital record won't help ensuring the authenticity of every step in the manufacturing process, therefore Everledger is not suitable for safety.

2.5.2 Origintrail

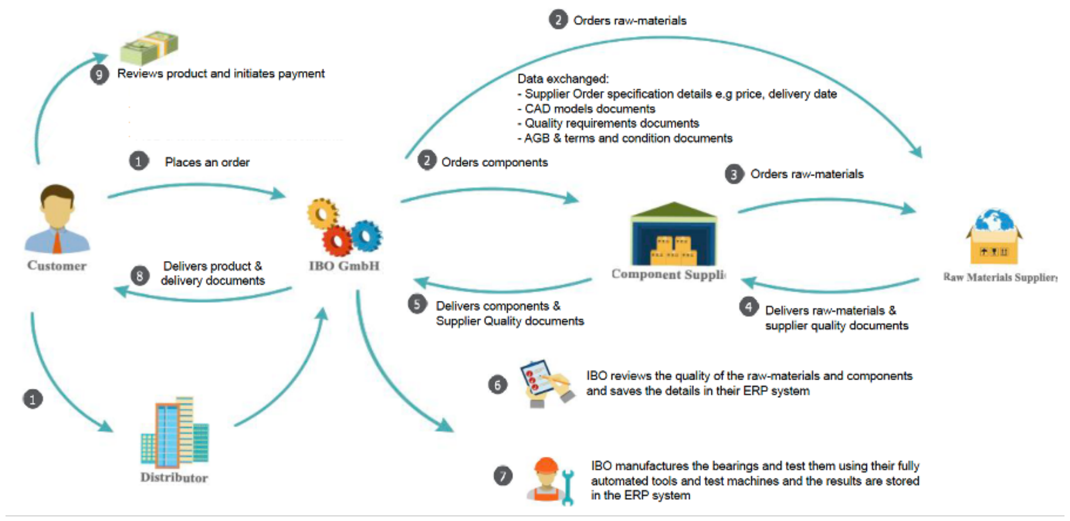
Origintrail is a blockchain-based ecosystem aimed at connecting legacy supply chain IT systems to ensure data immutability. The Origintrail protocol solves three significant issues: poor interoperability, inefficient data storage, reluctance to share data [32]. It is an Ethereum based system and uses ERC20 based tokens as incentives to track goods.

Like Everledger, Origintrail provides specialized solutions to track products and also, given Ethereum's weak points, in this thesis we decided to focus on a more generalized solution, not based on Ethereum, which can be applicable to various SMEs in the supply chain networ.

3 Research Approach

This chapter provides a detailed description of the supply chain process in IBO GmbH and relevant use-case scenarios, followed by the research methodology used to conduct this thesis.

3.1 Context



2 Orders raw-materials
9 Reviews product and initiates payment
Data exchanged:
- Supplier Order specification details e.g price, delivery date
- CAD models documents
- Quality requirements documents
- AGB & terms and condition documents
1 Places an order
2 Orders components
3 Orders raw-materials
Customer
IBO GmbH
Component Suppli
Raw Materials Supplier
8 Delivers product & delivery documents
5 Delivers components & Supplier Quality documents
4 Delivers raw-materials & supplier quality documents
6 IBO reviews the quality of the raw-materials and components and saves the details in their ERP system
1
Distributor
7 IBO manufactures the bearings and test them using their fully automated tools and test machines and the results are stored in the ERP system

Figure 3.1: IBO's supply chain process flow

IBO GmbH is a German company based in Munich dealing with manufacturing ball bearings from small to large bearings for high dynamic movements and complex environments [33]. They aim to provide services over their product's lifecycle - from manufacturing to maintenance [34]. Figure 3.1 describes the supply chain process in IBO.

IBO's sales team acquires the Customer orders through cold acquisition, networking, research, or support of existing customers and records the sales opportunity in the

Enterprise Resource Planning (ERP) system. The confirmation, along with the Requirements, CAD models, and terms & conditions documents, is dispatched to the Customer once the order is confirmed.

IBO initiates the requirements planning phase for raw material units, components, tools, equipment and assesses risks and opportunities. Once it finalizes the Suppliers, IBO places an order for the raw materials and the components. Both parties need to agree on the order specification details, including the price, expected delivery date, quality, etc. At this point, the Component Supplier orders the raw materials as well. Once the raw materials and components are ready, the Supplier ships them to IBO, including quality documents and additional material on request.

Upon receiving the materials, IBO reviews the quality of the shipment and verifies the documents. All details are saved in the ERP system, and, in case of incongruity, a complaint is raised to the Supplier.

IBO uses the raw materials and components to manufacture the ball bearings and tests using in-house, fully automated test machines. The details of the manufacturing and the results of the testing are also stored in the ERP system.

Once the order is ready, IBO labels the shipment, attaches the agreed documents, and dispatches it. The logistics company involved in the shipment is notified in case of special delivery conditions.

Upon receiving the shipment, the Customer inspects the product and the documents, and if satisfied, they initiate the payment; otherwise, a complaint is raised.

This thesis concentrates on using blockchain-based solutions for SMEs similar to IBO in safety-critical supply chains to provide traceable product history and transparent data flow without disclosing business secrets.

Transparent data flow

Figure 3.2 illustrates the inefficient audit process. Data transfer between the parties in the supply chain mostly occurs through emails and paper documents. In some cases one party might not save the documents sent with the shipment.

In the case of claims raised by the Customer, e.g., low quality of raw materials, IBO tries to resolve it by consolidating the data from various sources, including contacting other parties, which takes weeks. If the claims are not internally resolved, both parties need to agree on appointing a third-party auditor to settle the claims.

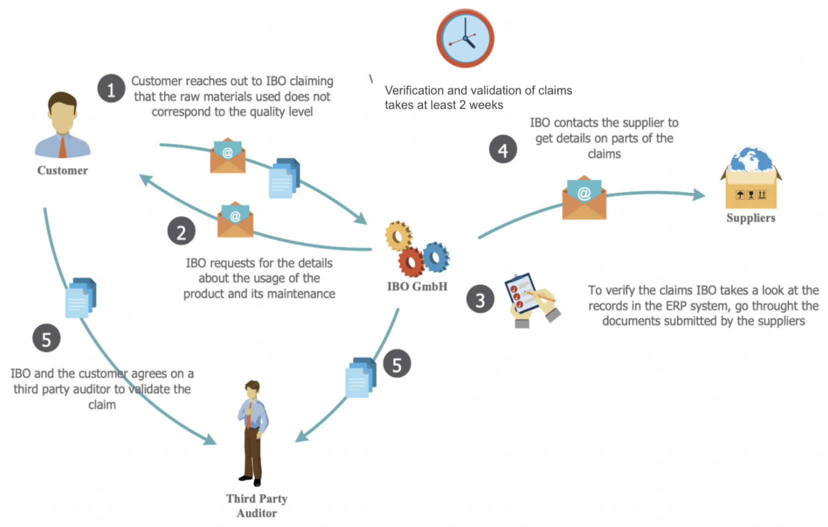

Customer
1 Customer reaches out to IBO claiming that the raw materials used does not correspond to the quality level
Verification and validation of claims takes at least 2 weeks
4 IBO contacts the supplier to get details on parts of the claims
Suppliers
2 IBO requests for the details about the usage of the product and its maintenance
IBO GmbH
3 To verify the claims IBO takes a look at the records in the ERP system, go throught the documents submitted by the suppliers
5 IBO and the customer agrees on a third party auditor to validate the claim
5
Third Party Auditor

Figure 3.2: Process flow showing the inefficient audit

This problem is caused by the lack of standards, the absence of a platform to share data, or by privacy issues, where the company does not want to disclose too much information. It becomes quite challenging for the Auditors and the Partner Companies to track, manage, and verify the data's authenticity because it might be distributed throughout various storages.

Blockchain-based technology allows data to be stored in a secure, distributed, tamper-proof system, where it becomes easy for partner companies to contribute, audit, and verify the authenticity of the data. It can act as an interoperable storage for supply chain data for parties with no mutual trust. The verification can be done quickly as the blockchain is accessible by all the parties having access rights, and the data is also secure from fraud as the ledger is immutable.

For this project, we represent a supply chain step as an *"Order"* which is a collection of data with an Order ID, stored in the ledger. An example of an Order is a purchase order placed by the Customer to the Manufacturer, with the data composed by the requirements, quotations, terms and condition documents, and the contract.

Product Tracking

A final product is the result of multiple steps in a supply chain, starting from raw materials to the finished assembled and certified piece. Because of several processes and parties involved, the supply chain is susceptible to fraud. In the scenario illustrated in Figure 3.3, the Distributor ends up selling a counterfeit product disguised as an IBO product. Because the counterfeit bearing is of lower quality than IBO´s, it results in a failure in the Customer’s system, which is blamed on IBO. These false claims could result in tremendous reputation loss which can lead to other Customers and Distributors in the system to lose faith in IBO.

In the blockchain network, all the related parties approve the data uploaded and eventual updates, preventing data tampering. The relevant parties can securely track the product using the traceable product history of the supply chain. For a ball bearing product the ideal traceable product history will be:

1. Ball bearing ordered to *Manufacturer* by *Customer*
2. Raw materials ordered to *Supplier 1* by *Manufacturer*
3. Component ordered to *Supplier 2* by *Manufacturer*
4. Raw materials extracted by *Supplier 1*
5. Raw materials shipped to *Manufacturer* by *Supplier 1*
6. Raw materials quality check by *Manufacturer*
7. *Manufacturer* pays *Supplier 1*
8. Component manufactured by *Supplier 2*
9. Components shipped to *Manufacturer* by *Supplier 2*
10. Components quality check by *Manufacturer*
11. *Manufacturer* pays *Supplier 2*
12. Ball bearing manufactured using raw materials and components by *Manufacturer*
13. Ball bearings quality tested by *Manufacturer*
14. Ball bearings shipped to *Customer* by *Manufacturer*
15. Ball bearings quality tested by *Customer*
16. *Customer* pays *Manufacturer*

This history is immutable and each step is legally bound to a contract between parties, so this ensures that the ball bearing is produced in the legitimate advertised way.

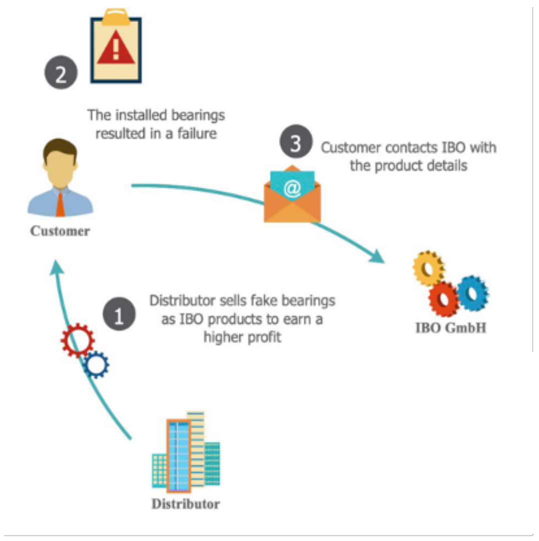

2
The installed bearings resulted in a failure
3
Customer contacts IBO with the product details
Customer
1
Distributor sells fake bearings as IBO products to earn a higher profit
IBO GmbH
Distributor

Figure 3.3: Process flow illustrating a dispute scenario

3.2 Design Science Research Methodology

Design Science Research Methodology (DSRM) is a research approach that comprises policies, methods, and procedures and consists of six steps: problem identification and motivation, defining the objectives for a solution, design and development, demonstration, evaluation, and communication [35]. Figure 3.4 below depicts the process model consisting of the six steps described later in this section.

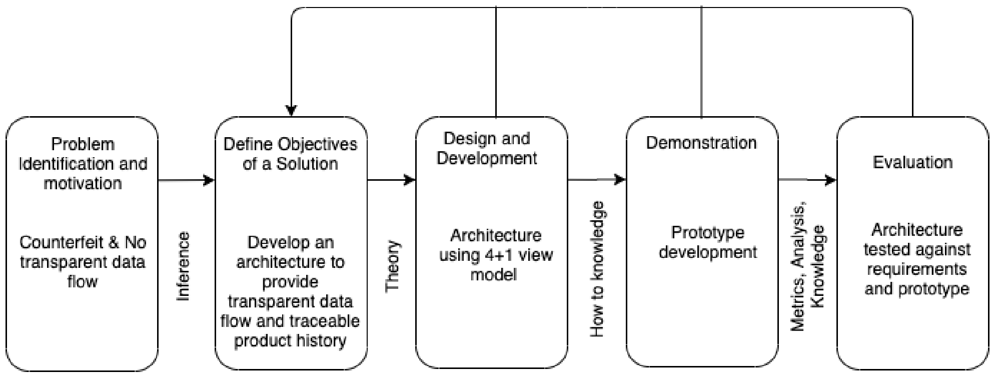

Problem Identification and motivation
Counterfeit & No transparent data flow
Inference
Define Objectives of a Solution
Develop an architecture to provide transparent data flow and traceable product history
Theory
Design and Development
Architecture using 4+1 view model
How to knowledge
Demonstration
Prototype development
Metrics, Analysis, Knowledge
Evaluation
Architecture tested against requirements and prototype

Figure 3.4: DSRM process model

3.2.1 Problem Identification and motivation

Counterfeit products are often of a lower quality standard, and when distributed, can result in safety risks ranging from severe to mild, reputation, and financial loss. At the moment, there is no way to trace the authenticity of a product. Another challenge is related to a transparent data flow. When the Customer reach out to the Manufacturer with claims regarding the products, e.g., the quality of the raw materials used, manufacturing tools, etc. verification and validation of such claims, if even possible, can take weeks, as most of the data needs to be consolidated from various sources, which are not fraud-proof.

3.2.2 Define the objectives for a solution

The objective is to build a blockchain-based single source, a fraud-proof traceable solution that enables users to track and trace their products and have a transparent data flow to help with faster and reliable claim processing. In a blockchain-based system, the transactions and the data states are transparent and available to all parties. One of the challenges in designing and implementing smart contracts considering the confidentiality and privacy of sensitive data. The first research question focuses on gathering and analyzing the requirements. The requirements are formulated using the template proposed by Pohl and Rupp, which is covered in the next chapter. Chapter 4 also includes the use-case scenarios and dependency diagrams.

3.2.3 Design and Development

The architecture for this thesis is defined using the 4+1 view modes considering different stakeholders' points of view. The models include the following views: (1) logical view consisting of the class diagram, (2) process view consisting of sequence and activity diagrams, (3) implementation view consisting of the package and component diagram, (4) deployment view consisting of deployment diagram and (5) scenario view consisting of user stories.

3.2.4 Demonstration

A prototype was implemented using Hyperledger fabric technology. It allows users to enter their product data that is confidential to their organization, share data with a different organization, and agree on the data shared between them. The user of an organization can access the history of a given product and request access to sensitive information that is not visible. The data stored in the private data collection is confidential and private to its own organization and cannot be accessed by other organizations in the system unless granted access. No private information is stored on the blockchain. Sensitive information can be shared between organizations using private data channels.

3.2.5 Evaluation

The architecture is evaluated against requirements and the prototype implemented. Assessed scenarios are related to traceable product history and transparent data flow.

3.2.6 Communication

The activity is not discussed here, as it has not been published in a paper or any academic journal yet.

4 Requirement and System Analysis

This chapter tackles the first research question. The first section consists of the stakeholder requirement described for the supply chain system, followed by the detailed use-case analysis. The chapter concludes with the comparison between a traditional supply chain system and a blockchain-based system.

4.1 Stakeholder Analysis

This section focuses on identifying stakeholders and defining the relationship between them. Edward Freeman defines stakeholders as any group or individual who can affect the organization by achieving its objectives or be affected in the process [36]. The goal here is to identify the stakeholders and their dependencies by analyzing different parties in IBO's supply chain.

The main stakeholders identified for this thesis are as follows: Customers, Manufacturers, Raw Material Suppliers, Component Suppliers, and Distributors. Customers are the catalysts for most of the activities in the supply chain network. There would be no need for a supply chain if there was no customer demand. Manufacturers are the producers of goods and services that are useful to end-users, and they help transform components and raw materials supplied by Suppliers into beneficial products [37]. Suppliers can be cataloged into a wide range of categories known as tiers, including raw materials producers or extractors, Components Suppliers, Components Assembler, and various other entities. Tier 1 supplier, e.g., Component Suppliers, cater to the Manufacturers' needs, and Tier 2 Suppliers, e.g., Raw Material Suppliers, support the Tier 1 Suppliers [38]. Distributors pay an important role in the supply chain by acting as a mediator between the Customers and the Manufacturers.

Dependencies between the stakeholders are a way to understand the actors' behavior in the given context [39]. Figure 4.1 above depicts the dependency network diagram, which helps identify the resources required by the stakeholders and their dependencies. Using the approach mentioned in the study of Balta, Greger, Wolf, and Krcmar (2015) [39], five main actors are identified. Each actor is represented by his activities and goals,

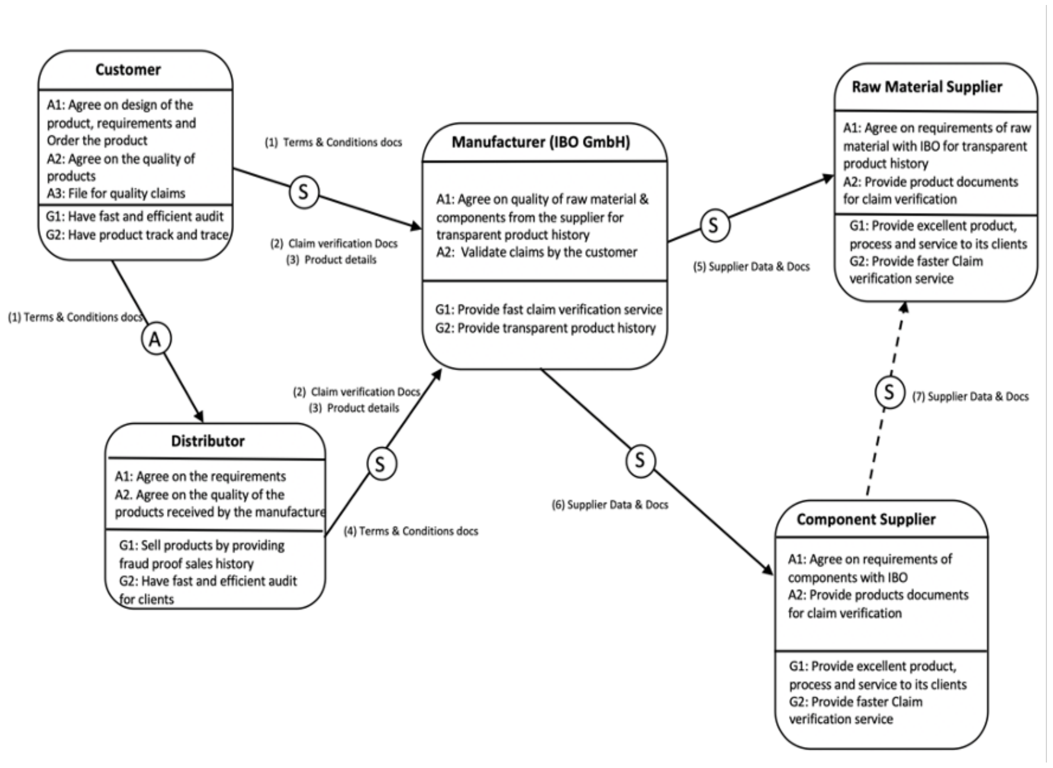

Customer
A1: Agree on design of the product, requirements and Order the product
A2: Agree on the quality of products
A3: File for quality claims
G1: Have fast and efficient audit
G2: Have product track and trace
(1) Terms & Conditions docs
(2) Claim verification Docs
(3) Product details
Manufacturer (IBO GmbH)
A1: Agree on quality of raw material & components from the supplier for transparent product history
A2: Validate claims by the customer
G1: Provide fast claim verification service
G2: Provide transparent product history
Raw Material Supplier
A1: Agree on requirements of raw material with IBO for transparent product history
A2: Provide product documents for claim verification
G1: Provide excellent product, process and service to its clients
G2: Provide faster Claim verification service
(5) Supplier Data & Docs
(1) Terms & Conditions docs
A
S
(7) Supplier Data & Docs
Distributor
A1: Agree on the requirements
A2. Agree on the quality of the products received by the manufacture
G1: Sell products by providing fraud proof sales history
G2: Have fast and efficient audit for clients
(2) Claim verification Docs
(3) Product details
(4) Terms & Conditions docs
(6) Supplier Data & Docs
Component Supplier
A1: Agree on requirements of components with IBO
A2: Provide products documents for claim verification
G1: Provide excellent product, process and service to its clients
G2: Provide faster Claim verification service

Figure 4.1: Interactions and dependencies between stakeholders

and a dependency is depicted using a directional arrow starting at the dependent actor and ending at an independent actor [39]. The concept of power is shown as a circle on the dependency arrow, and it represents the control of the resource, which could be of two types: symmetric and asymmetric [39]. In Figure 3.1, five actors and six dependencies are illustrated, one of which is asymmetric, and another one represented by a dashed arrow is a secondary dependency.

The Customer has to agree with the requirements (activity A1 of Customer) and the terms and conditions while placing an order for a product with the Manufacturer or the Distributor. To track the progress or the authenticity of the product (goal G2 of Customer), the Customer depends directly or indirectly - through the Distributor - on the Manufacturer for the order details (dependency 3 "order details"). Similarly, the Raw Material and Component Supplier need to agree on requirements and terms & conditions (activity A1 of the Raw material and component supplier) while accepting an order by the Manufacturer.

In the case of an issue, the Customer creates a claim, e.g., regarding the quality of raw materials used to the Manufacturer and expects fast processing (goal G1 Customer). To provide a fast claim verification process (goal G1 of Manufacturer), the Manufacturer is dependent on the supplier data and documents (dependency 5 and 6 "Supplier data & docs") coming from the Raw Material and Component Supplier. The documents provided by the Component supplier (dependency 6 "Supplier docs") could be generated by the component supplier itself or could be provided by the Raw Material Supplier (dependency 7 "Supplier docs").

Since most stakeholders have similar goals and perform analogous actions, we abstract them into two roles: Consumers and Producers. An organization in the network at each step of the supply chain assumes either role. In a simple supply chain consisting of a Customer, a Manufacturer, and a Supplier, during the transaction between the Customer and the Manufacturer, the Customer acts as the Consumer, and the Manufacturer serves as the Producer. During the transaction between the Manufacturer and the Supplier for the Raw Materials, the Manufacturer acts as the Consumer, and the supplier acts as the Producer.

4.2 Use case Analysis

In this section, we cover the detailed use-case analysis for system and business requirements, which are summarized in Table 4.1.

4.2.1 System Requirements

The system developed for this project is blockchain based and should maintain an immutable list of past transactions. Transactions are records of events that occur in the blockchain with a given timestamp and are generated using smart contracts. Smart contracts are business code that can modify the state of the ledger. Only authorized parties in the network can access the ledger and execute the smart contracts using API calls.

4.2.2 Business Requirements

The Supply chain system is divided into two modules, the *Transparent Data Flow* module and the *Product Tracking* module. Both modules have their own ledger and smart contracts.

Transparent Data Flow

This module deals with *Orders* and operations on those. Such module is required to be transparent, trusted, and fraud-proof in order to guarantee a fast and efficient audit. Its ledger stores the details, the reviews and approval of the order. Consumers can upload the order details and tag the Producer for approval. At any given time, the Producer can choose to update the order details or approve them. In case the Producer updates the order details, the Consumer needs to approve the changes as well. Figure 4.2 illustrates the use-case diagram for this module.

External members of the network can request access which, if granted, allows them to view the Order details of a certain step in the supply chain. This ties to the requirement for data privacy and confidentiality in the context of data protection and security, which is concerned with business secrets and how data is shared with third parties.

Moreover the module is required to follow GDPR , which deals with the rights of an individual or organization to have control over confidential information. General Data Protection Regulation (GDPR) is Europe’s privacy and security laws passed by the European Union (EU) with the intent to protect EU organizations and citizen’s private information. Organizations that don’t comply with the policies are heavily fined, which could be up to 4% or the annual global turnover or 20 million euros, whichever is higher [40]. There have been many discussions related to the effects of GDPR on Blockchain. The GDPR policy states that the data controller is a natural or legal person that should comply with the regulations [41]. However, blockchain is distributed and does not have a responsible central authority. Secondly, Article 17 of the GDPR policies states, "Right

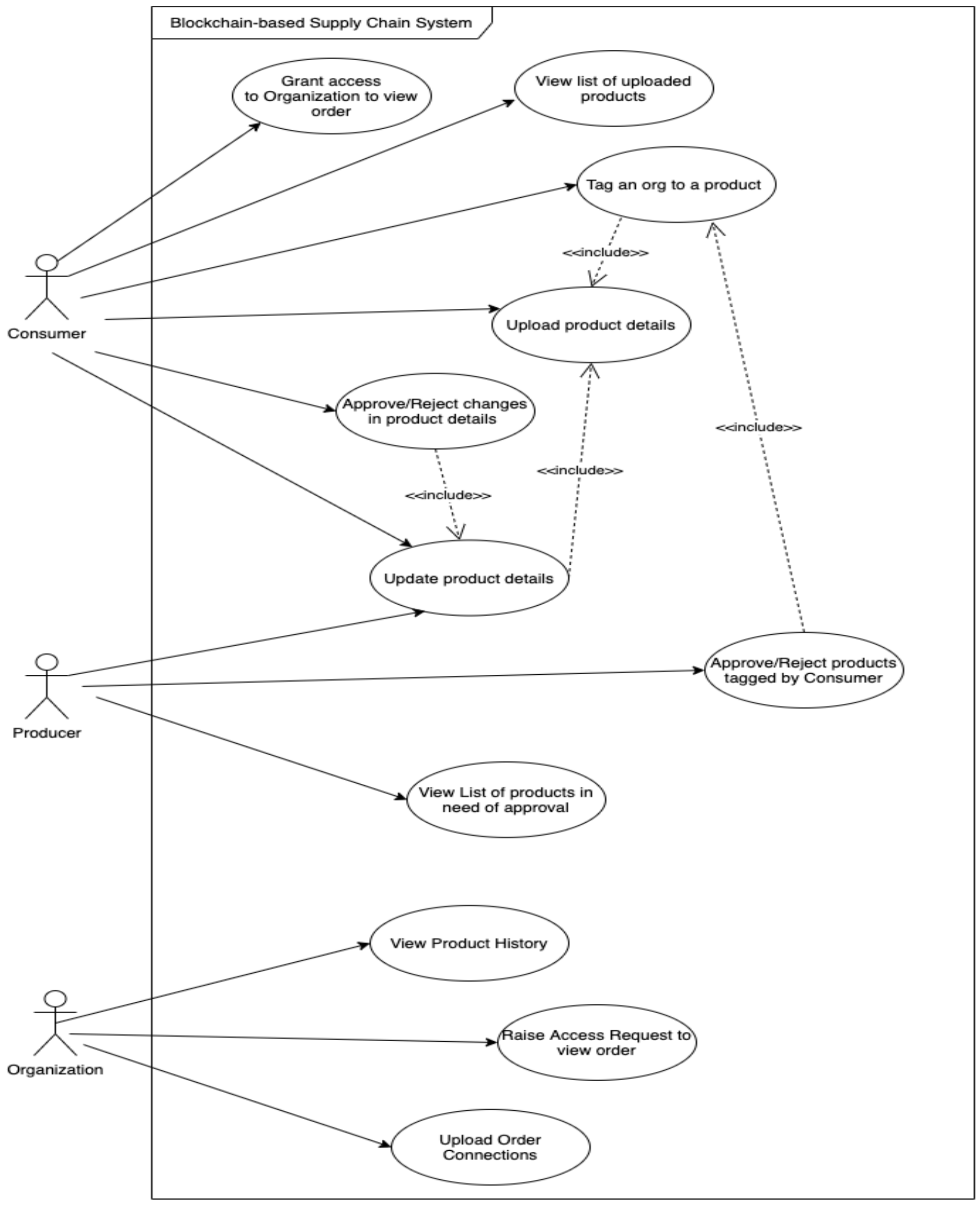

Blockchain-based Supply Chain System
Grant access to Organization to view order
View list of uploaded products
Tag an org to a product
<<include>>
Consumer
Upload product details
Approve/Reject changes in product details
<<include>>
<<include>>
<<include>>
Update product details
Producer
Approve/Reject products tagged by Consumer
View List of products in need of approval
View Product History
Organization
Raise Access Request to view order
Upload Order Connections

Figure 4.2: Use case diagram for Blockchain-based Supply Chain System

to erasure (Right to be forgotten)," i.e., the data can be erased whenever necessary [41]. However, once the transaction is uploaded on a blockchain, it cannot be modified or deleted.

In this thesis, the organizations that upload the data are considered data controllers, which means they have the right to update and share data with any other organization in the network (upon Access Request) and, if necessary, delete it from the network.

Product Tracking

This module is responsible for creating and maintaining the Traceable Product History. Products can be single items or batches of items which are traceable by an ID. The Consumer can access the Product History of any product in the blockchain network using such id. This is achieved by inspecting the connections between orders. Since an order is associated to a Producer and a Consumer, a connection is a link between two orders where the Producer of Order 1 is the Consumer of Order 2. For example, given Order 1 "Manufacturer orders component by Supplier " and Order 2 "Supplier orders raw material by Extractor" then a Connection 1 can be made between Order 1 and Order 2 by the Supplier. If the Order does not involve a partner, the single agent participating in the Order is both Producer and Consumer. Starting from the Order ID the module extracts all connections to build the Product History.

Due to the privacy regulations mentioned before, only a subset of manufacturing details stored in the blockchain of the *Transparent Data Flow* module are reflected (copied) in the blockchain of this module.

4.3 System Analysis

The goal of System analysis is to create a new and improved system by examining current processes and workflows [42]. The requirements mentioned above are compared with traditional supply chain system and the differences listed in Table 4.2.

In a traditional supply chain system, each organization in the chain stores the data in its Enterprise Resource Planning (ERP) system. The data communicated between both parties is duplicated in their respective ERP systems and could be inconsistent. Moreover, the tracking data needs to be consolidated from various sources.

	As a. . .	Id	I want/need to ...
System Requirements	System	R1	Record user actions and store them in the ledger
		R2	Maintain an immutable list of transactions in the ledger
		R3	Add mechanism to limit access to the ledger
		R4	Expose the smart contracts using APIs
		R5	Authorize users to execute smart contracts
Transparent data flow Requirements	Consumer	R6	Create new order
		R7	Update order details
		R8	Tag the Producer for approval
		R9	Approve/Reject order details changes from the Producer
		R10	View a list of all orders
		R11	Restrict access to the order uploaded only to the Producer
		R12	Restrict access to the order shared only to the authorized organizations
		R13	Delete the order.
		R14	Approve order access request
		R15	View list of order access requests that need approval
	Producer	R16	Approve/Reject order details agreement
		R17	Update the order details
		R18	View a list of orders that need approval
	Organization	R19	Raise an access request to view the details of an order.
		R20	View the list of raised access requests.
Product Tracking Requirements	Organization	R21	Access the product history using the order id.
		R22	Upload order connections

Table 4.1: Requirements of a blockchain-based supply chain system

	Traditional Supply chain	Blockchain-based supply chain
Data Storage	Each party stores its own data and the data that is shared by the other parties.	Each data record is stored on the ledger upon which the parties had previously agreed.
Data Sharing	Data is shared through emails, paper-documents.	Any party can access the data only if granted authorization by the owner of the data.
Audit Time	The process can be lengthy (up to several weeks) because the validation is done on each party's data copies.	The validation can be quickly performed by accessing the ledger as the parties agree upon the data stored.
Product Tracking	To track each phase in the supply chain, partial data from all party members must be processed and assembled into a product history. As more members are added to the supply chain, the time to complete this process increases exponentially.	The product history can be built by traversing the product connections in the ledger.
Fraud	Party members can compromise their own data and update history leading to a false product history.	The data update history is stored on the ledger, which is tamper-proof by design.
Security	Traditional supply chains are vulnerable to data breaches and cyber attacks.	As the blockchain is replicated on all the nodes in the network, the consensus mechanism and the cryptographic techniques, cyber attacks are almost impossible.

Table 4.2: Difference between traditional supply chains and block-chain based supply chains

5 System Design

This chapter describe the architecture in the first section followed by a detailed explanation on how to retrieve Product History.

5.1 4 + 1 Architectural Views

Architecture and design play an essential role in shaping the system. They comprise components, their relationship with each other, the guidelines governing their design and evolution over time [43]. A multi view architecture is a model composed of multiple perspectives allowing us to address the concerns of various stakeholders in the system. One such modeling technique is the 4 +1 architectural view model [9], and it provides four essential views, each addressing a specific stakeholder concern and is illustrated in Figure 5.1.

5.1.1 Logical View

The Logical view describes the functional requirements of the system. The system is decomposed into classes and objects, which are abstractions taken from the problem domain [9]. UML class diagrams are used to describe the logical architecture. System analysts and designers are the stakeholders who are interested in the logical view of the system.

The system's architecture is divided into two main modules: the *Transparent Data Flow* module and the *Product Tracking* module.

Transparent data flow

Figure 5.2 illustrates the class diagram for the smart contracts of the *Transparent Data Flow* module and is responsible for creating, updating and accessing orders entered by the organization. Described below is a detailed explanation of the different classes used in the order details module.

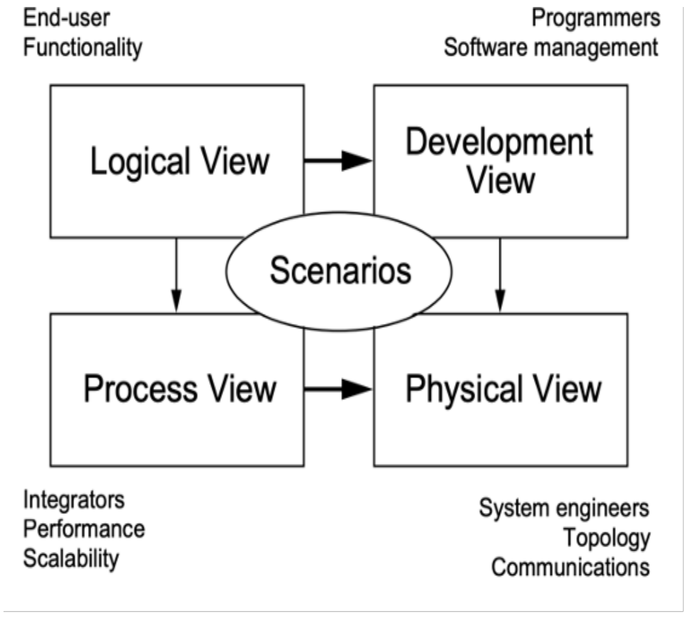


Figure 5.1: 4+1 View Model

1. **OrderStatus:** represent the status of the order created by the organization. When an organization creates a new order, the status is set to PENDING or NOT_NEEDED, depending on the approverOrganization. The status of the order can be updated using the approveOrderStatus function in the smart contract, which changes the order to APPROVED or REJECTED.
2. **File:** represents the file uploaded by the organization that is tied to an order. The File contains the content's hash, the name of the file, and the timestamp. An Organization can upload multiple files in a single order.
3. **Comment:** represents the comments added by the organization while creating, updating, or approving the order. It consists of the comment string, timestamp, username, and organization name. Similar to the File, an order can have multiple comments.
4. **Order:** represents the order created by the organization, which describes a process step in the supply chain, e.g., Sales Order, Manufacturing process, Purchase order, etc. When an organization creates a new order, it needs to mention the order id, type, and order date. If an order has the approver organization name stated, then the approver organization can access and update this order too.
5. **OrderContract:** represents the smart contract responsible for creating, updating,

viewing, and deleting the order details. The smart contracts are developed using the fabric-contract-API in Java.

6. **AccessRequestState:** represents the state of the request, which could be PENDING, APPROVED, or REJECTED.
7. **AccessRequest:** represents a request raised by a user from an organization to access the details of a particular order.
8. **AccessRequestContract:** is the smart contract responsible for creating, viewing, updating, or deleting these access requests.
9. **PrivateDataCollection:** is a data class representing the private data between two organizations, identified by the policy field which contains the name of the organizations.
10. **PrivateDataCollectionUtils:** helper class for PrivateDataCollection.

Product Tracking:

The *Product Tracking* history module is responsible for providing product track and trace to the end-user. The product history is created based on the Connections and is displayed to the user as a set of Orders consisting of minimal data such as the date or the organizations names that handled the Order.

Figure 5.3 depicts the class diagram for the product history module, followed by a detailed description of the classes presented.

1. **OrderDetails:** represents a step that would be presented when the user accesses a product's history. The data displayed to the user includes the order id, type, name of the publishing organization, and the date.
2. **ProductHistoryConnection:** represents the connection between two orders.
3. **ProductHistoryContract:** represents the smart contract that handles the creation and deletion of these connections and the viewing of the product history.

5.1.2 Process View

The system's process view represents the dynamic aspect that takes the non-functional requirements into account, e.g. performance [9]. The architecture can be described using several abstraction levels, each addressing a different interest [9]. This can range from high level independent logical networks to basic jobs running in one process. UML

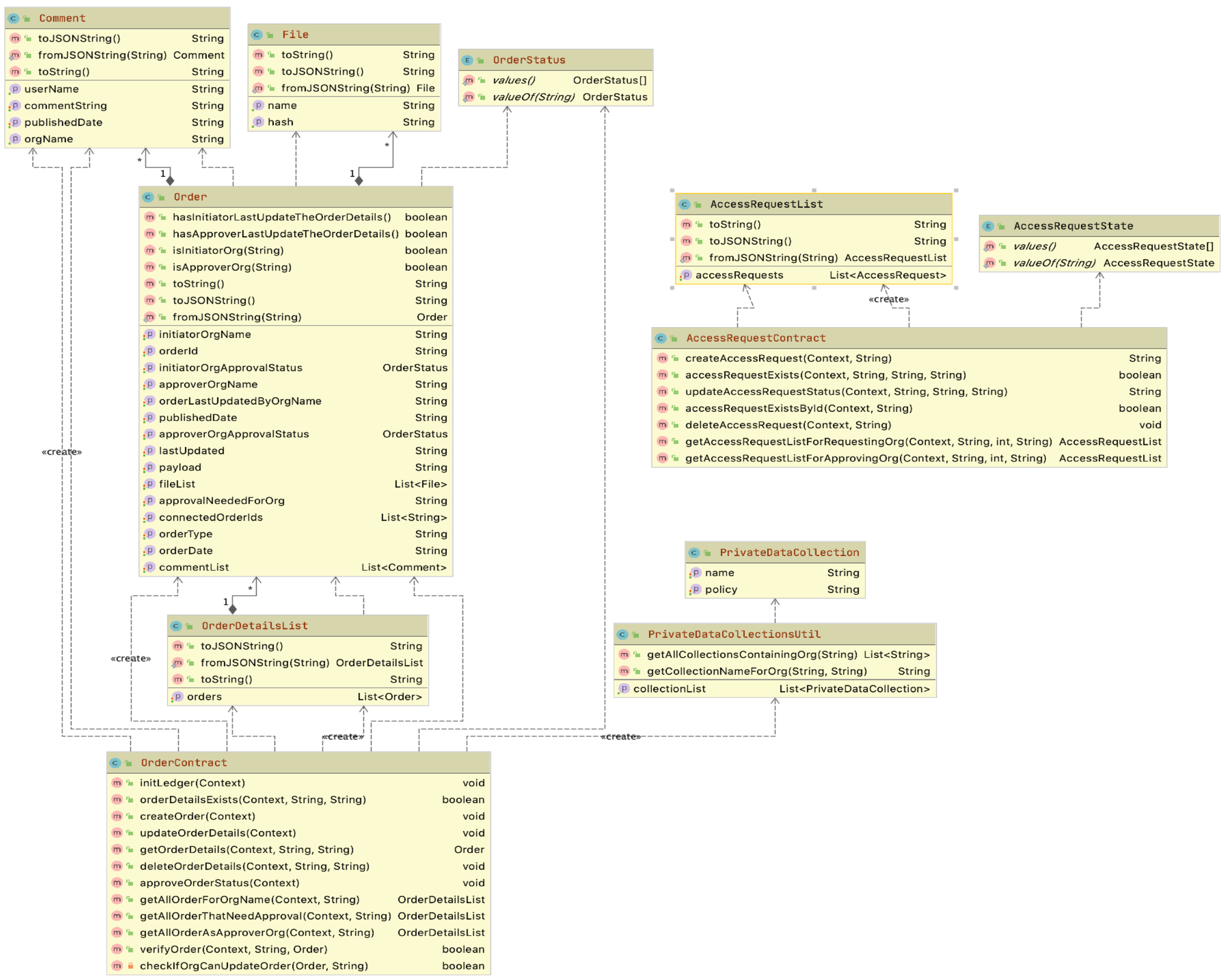

Comment
toJSONString() String
fromJSONString(String) Comment
toString() String
userName String
commentString String
publishedDate String
orgName String
File
toString() String
toJSONString() String
fromJSONString(String) File
name String
hash String
OrderStatus
values() OrderStatus[]
valueOf(String) OrderStatus
Order
hasInitiatorLastUpdateTheOrderDetails() boolean
hasApproverLastUpdateTheOrderDetails() boolean
isInitiatorOrg(String) boolean
isApproverOrg(String) boolean
toString() String
toJSONString() String
fromJSONString(String) Order
initiatorOrgName String
orderId String
initiatorOrgApprovalStatus OrderStatus
approverOrgName String
orderLastUpdatedByOrgName String
publishedDate String
approverOrgApprovalStatus OrderStatus
lastUpdated String
payload String
fileList List<File>
approvalNeededForOrg String
connectedOrderIds List<String>
orderType String
orderDate String
commentList List<Comment>
AccessRequestList
toString() String
toJSONString() String
fromJSONString(String) AccessRequestList
accessRequests List<AccessRequest>
AccessRequestState
values() AccessRequestState[]
valueOf(String) AccessRequestState
AccessRequestContract
createAccessRequest(Context, String) String
accessRequestExists(Context, String, String, String) boolean
updateAccessRequestStatus(Context, String, String, String) String
accessRequestExistsById(Context, String) boolean
deleteAccessRequest(Context, String) void
getAccessRequestListForRequestingOrg(Context, String, int, String) AccessRequestList
getAccessRequestListForApprovingOrg(Context, String, int, String) AccessRequestList
PrivateDataCollection
name String
policy String
PrivateDataCollectionsUtil
getAllCollectionsContainingOrg(String) List<String>
getCollectionNameForOrg(String, String) String
collectionList List<PrivateDataCollection>
OrderDetailsList
toJSONString() String
fromJSONString(String) OrderDetailsList
toString() String
orders List<Order>
OrderContract
initLedger(Context) void
orderDetailsExists(Context, String, String) boolean
createOrder(Context) void
updateOrderDetails(Context) void
getOrderDetails(Context, String, String) Order
deleteOrderDetails(Context, String, String) void
approveOrderStatus(Context) void
getAllOrderForOrgName(Context, String) OrderDetailsList
getAllOrderThatNeedApproval(Context, String) OrderDetailsList
getAllOrderAsApproverOrg(Context, String) OrderDetailsList
verifyOrder(Context, String, Order) boolean
checkIfOrgCanUpdateOrder(Order, String) boolean
«create»
«create»
«create»
«create»
«create»
1
*
1
*
1
*

Figure 5.2: UML class diagram for Transparent data flow

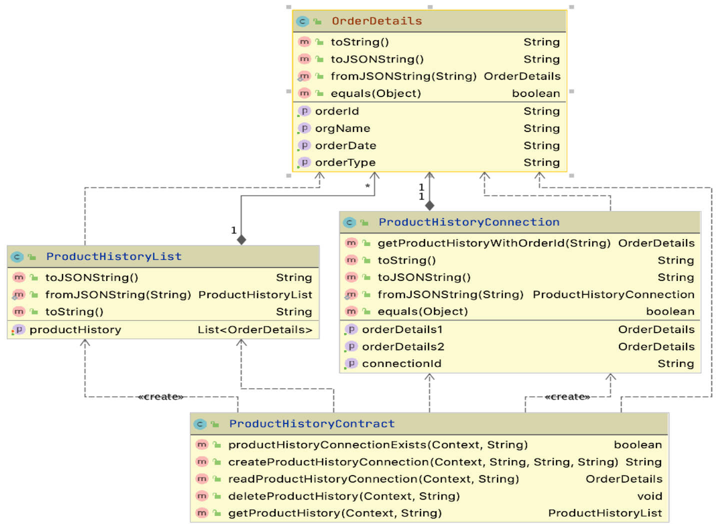

OrderDetails
toString() String
toJSONString() String
fromJSONString(String) OrderDetails
equals(Object) boolean
orderId String
orgName String
orderDate String
orderType String
ProductHistoryList
toJSONString() String
fromJSONString(String) ProductHistoryList
toString() String
productHistory List<OrderDetails>
ProductHistoryConnection
getProductHistoryWithOrderId(String) OrderDetails
toString() String
toJSONString() String
fromJSONString(String) ProductHistoryConnection
equals(Object) boolean
orderDetails1 OrderDetails
orderDetails2 OrderDetails
connectionId String
ProductHistoryContract
productHistoryConnectionExists(Context, String) boolean
createProductHistoryConnection(Context, String, String, String) String
readProductHistoryConnection(Context, String) OrderDetails
deleteProductHistory(Context, String) void
getProductHistory(Context, String) ProductHistoryList
«create»
«create»
1
*
1
1

Figure 5.3: UML class diagram for Product Tracking module

sequence, activity, or communication diagram are used to represent the architecture in the process view. Sequence diagrams illustrate the interactions in sequential order, and Business Process Model and Notation (BPMN) helps to specify business processes in a graphical representation [44]. Section 5.2.1 describes the sequence diagrams designed for this thesis, and section 5.2.2 focuses on the BPMN diagrams.

Sequence Diagram

The sequence diagrams for the order creation is illustrated in Figure 5.4. The components involved in creating a new order are the organization (e.g., Manufacturer), the app, a distributed file storage, the smart contracts, smart contracts' APIs, and the distributed ledger. The order id mentioned in the connection's list is used to create the product history.

1. **Step 1:** The user from a supply chain organization provides the necessary details, files, and the connection meta-data (containing order id mapping to another order in the system)
2. **Step 2-3:** All the files provided by the user are added to the distributed file storage, and the hash is returned.

3. **Step 4-6:** The order ids mentioned in the connections list are evaluated to check if they exist in the system.
4. **Step 7-9:** For all the valid order ids, create a connection in the product history ledger using the smart contract.
5. **Step 10-12:** Combining all the details acquired in the previous steps, the system submits a request to create an order using the smart contract.
6. **Step 13:** On successful response, display the newly created order details.

The sequence diagram for the other uses cases is similar and contains the same components therefore are not illustrated.

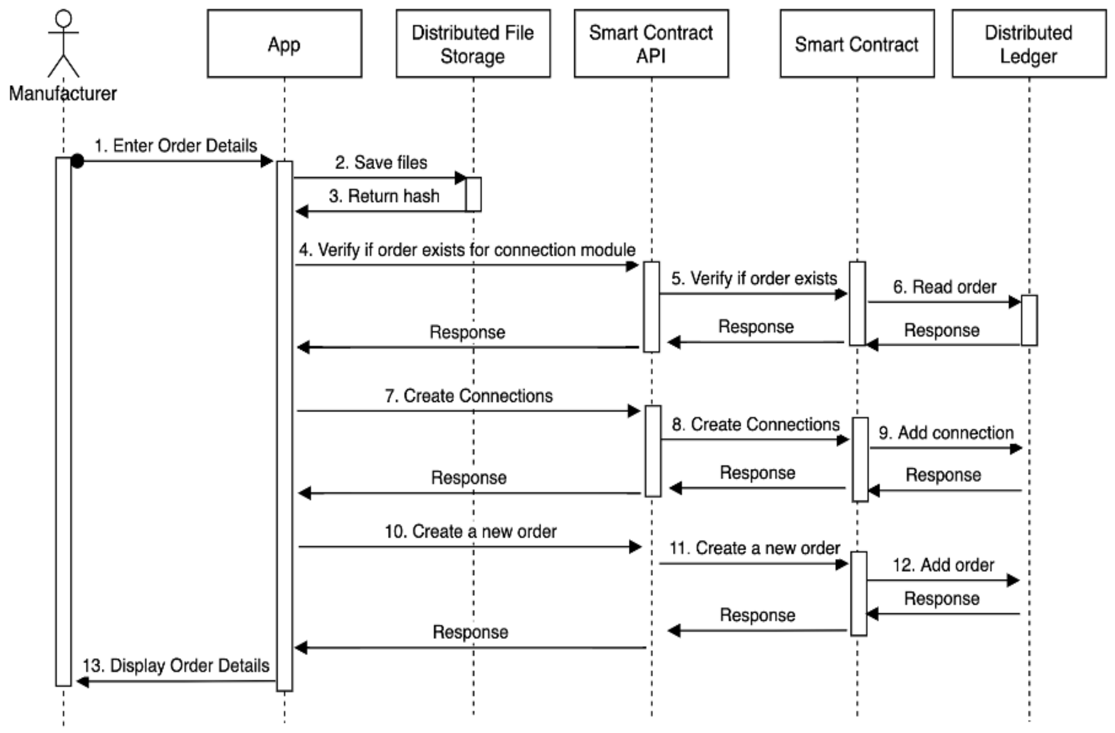


Figure 5.4: UML Sequence diagram for new order creation

Business Process Model and Notation (BPMN)

BPMN's goal is to provide notations easily understandable by all project stakeholders, from requirement analysts to developers [44]. The diagram consists of Flow objects, connecting objects, swimlanes, and artifacts:

1. Flow objects contain activities, events, and gateways [44]
2. Swimlanes are graphical containers that group the participant's actions
3. Connecting object are used to connect Flow objects
4. Artifacts consist of data objects, annotations, and groups and are not illustrated in the diagrams [44]

Figure 5.5 illustrates the business process diagram for a *Transparent Data Flow* scenario between three supply chain parties. The events in orange boxes represent the actions taken out of the system.

The Customer places an order for the roller bearings by contacting the Manufacturer. Upon receiving the order request, the Manufacturer creates a new order in the blockchain and simultaneously places an order for raw materials. After adding all the details and the required documents in the newly created order, the Manufacturer adds the Customer to review it. The Customer can access the order by viewing it on the list of orders to approve.

The Raw Material Supplier also creates a new order by adding all the necessary details and tags for the Manufacturer to approve. In the order approval process, which takes place between the Manufacturer and the Raw Material Supplier, the Manufacturer reviews the order. If everything is as promised, it approves the order or updates it and sends it back to the raw material Supplier to approve the changes. The Supplier follows the order approval process as well.

Upon receiving the Supplier order, the Manufacturer links it to the Customer order. This step helps in retrieving the details related to product history for the *Product Tracking* module.

5.1.3 Development View

The development view of the architecture comprises components and interfaces used to assemble a system. The system is packaged into small elements, e.g., libraries, extensions, or sub-systems that can be developed and tested by the independent development team. Component diagrams are implementation views that describe the components and their interactions with one another. Components are defined as independent units in a system. Figure 5.6 depicts the component diagram for a blockchain-based supply chain system.

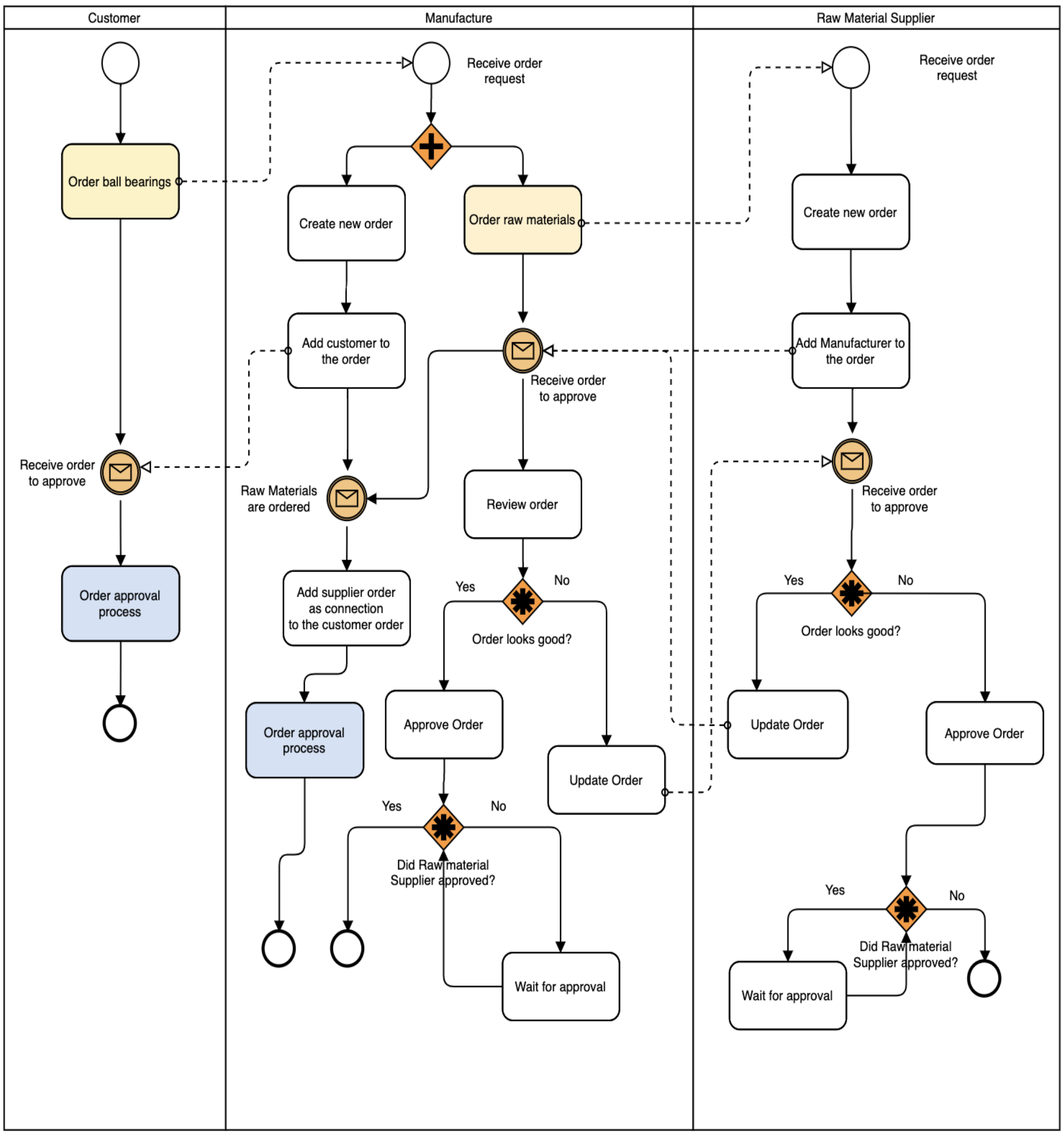


Figure 5.5: BPMN diagram for transparent data flow module

Listed below is the list of components with a detailed explanation of their interfaces and dependencies.

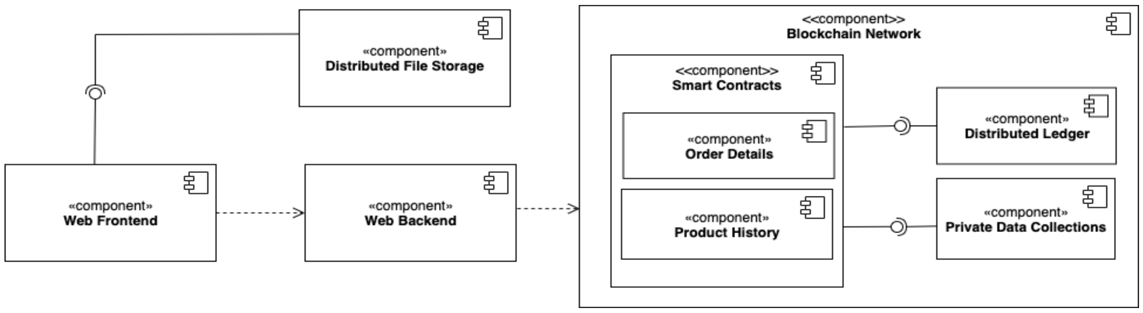


Figure 5.6: UML component diagram for blockchain-based supply-chain system

1. **User Interface:** The user interface is web-based and the user interacts using a web browser. The web interface provides various functionalities to the user to communicate with the blockchain network. The module depends on the Web backend REST API to request or update the data. The front end depends on the Distributed File System module to upload the files and retrieve the files' hash.
2. **Web Backend:** The Web backend provides APIs which allow the User Interface to communicate with the smart contracts. This component uses the fabric-client SDK to interact with the hyperledger fabric network. The user can perform API calls after obtaining an authorization token.
3. **Blockchain:** Blockchain contains three modules: the smart contracts, the distributed ledger, and the private data collections. We use hyperledger fabric to implement the blockchain module.
4. **Smart contracts:** The smart contracts are deployed on the nodes in the hyperledger network. It is the core of the project that handles most of the business logic. The private, confidential data in the network are stored in Private data collection, and the rest is stored in the world state of the distributed ledger.
5. **Distributed Ledger:** The distributed ledger is a journal of transactions that take place in the network. The ledger has two parts: the world state and the blockchain. The world state is a key-value pair database and is implemented using CouchDB. The blockchain here is a transaction log, and it records all events leading to the current world state.
6. **Private Data Collections:** Private data collections are used for confidential data and cannot be stored on the public chain. It consists of the actual data and the

hash of the data, which is stored on every organization's ledger, and it is used for state validation.

7. **Distributed File System (IPFS):** This component is responsible to upload the files on multiple nodes.

5.1.4 Physical View

In simple terms, the physical view can be described as the mapping from software to hardware [9] with the goal to deliver a deployable system. The physical view focuses on non-functional requirements such as availability, scalability, reliability, etc. Figure 5.7 illustrates the physical view in the form of a deployment diagram.

The User Interface is available via the Browser and it depends on the Web Frontend server to display the content. The Web Backend component is deployed on a server and communicates with the Blockchain node. The Smart Contracts, the Distributed Ledger and the Private Data Collections are deployed on the Blockchain Node and each organization in the network runs its own Blockchain Node.

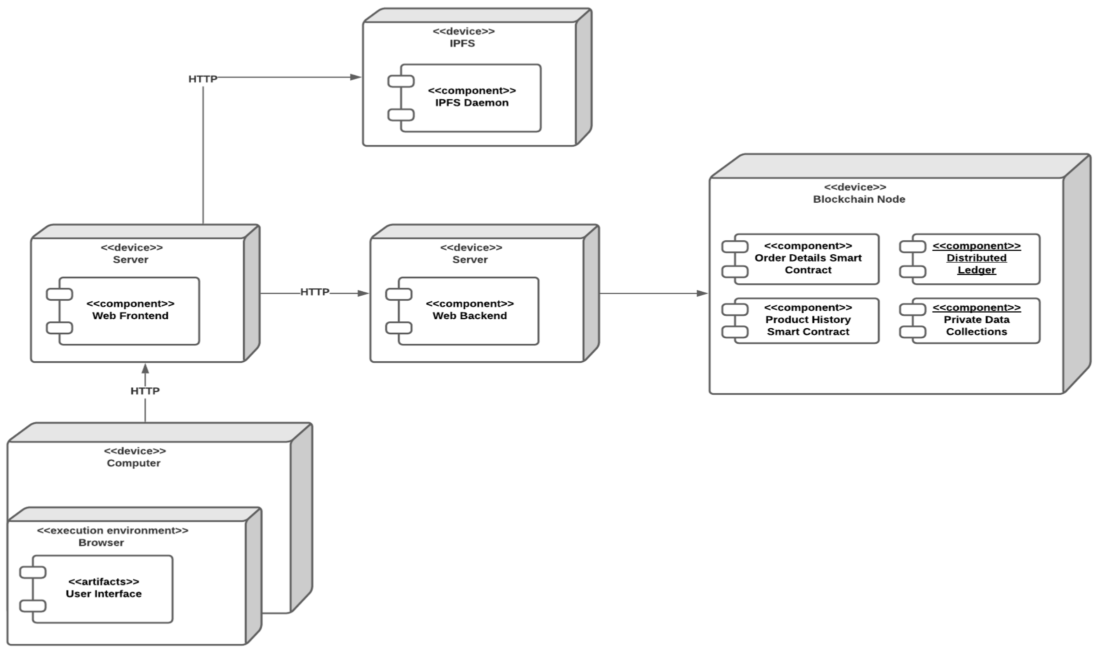


Figure 5.7: UML deployment diagram for blockchain-based supply-chain system

5.2 How to retrieve the Product History

In blockchain systems where the end-product is the only asset (see 2.5.1) retrieving a product history is trivial because the only state change is the ownership of the product, which means the product history is simply the list of owners in time. However in our system the Orders are distributed among several Private Data Collection which are only accessible by authorized organizations, which are a subset of the parties in the network. Because the end product is not necessarily the only asset, it is necessary to inspect the data present in all organizations involved in that product. Because of this from the point of view of each party, the Order information is always incomplete. In this thesis we tackled this problem in the following way. The ledger of the Product History stores a Connection, which builds a bridge between related orders in different organizations, even if these organizations do not share private data or don't even know each other. For example, the Customer places an order to the Manufacturer who then places an order to the Supplier, which is not visible to the Customer. It is the Manufacturer's duty the link the Customer order to the Supplier order in a connection and store it in the ledger. The complete Using the connections we adopt the following algorithm to build a list of chronological Orders describing the Product History.

Algorithm 1 Retrieve Product History

```
 1: procedure GetProductHistory($orderId$)
 2:     Create empty list $analyzedConnections$
 3:     Create empty list $productHistory$
 4:     Retrieve all connections in the ledger with $orderId$ and store them in
    $tempConnections$
 5:     for $connections$ in $tempConnections$ do
 6:         if $connections$ not present in $analyzedConnections$ then
 7:             Add $connection$ to $analyzedConnections$
 8:             Get $order1$ from $connection$
 9:             if $order1$ not present in $productHistory$ then
10:                 Add to $productHistory$
11:                 Goto step 4 for $orderId$ of $order1$
12:             Get $Order2$ from $connection$
13:             if $orde21notpresentinproductHistory$ then
14:                 Add to $productHistory$
15:                 Goto step 4 for $orderId$ of $order2$
16:     Sort $productHistory$ by timestamp
17:     if $orderId \neq productHistory[0]$ then
18:         Delete all records younger than the record with $orderId$
19:     Retrun $productHistory$
```

6 Prototypical Implementation and Evaluation

This chapter describes the technology stack used to implement the prototype for this thesis followed by the evaluation of the prototype based on the requirements and the reference architecture.

6.1 Tools and Technologies

6.1.1 Hyperledger Fabric

Hyperledger is an open-source community focused on creating frameworks, tools, and libraries for enterprise blockchain developments [45]. It includes various distributed ledger frameworks such as Fabric, Indy, Iroha, Sawtooth Burrow, etc. [45]. Founded in 2015, Hyperledger Fabric is a blockchain technology consisting of ledgers, smart contracts, and tools to help users manage transactions [22]. It has a modular architecture and allows the components to be used in a plug and play fashion. This feature covers a wide range of industry use cases. Fabric is private and permissioned. The smart contracts can be developed in Go, Java, or node.js programming languages. Fabric uses dockers to store images for the containers for various fabric components.

6.1.2 Docker

Docker is an open-source container technology that allows developers to package applications with the dependent libraries [46]. The docker tools are designed to help developers create, run, and deploy applications quickly. The developers need not worry about the system environment that the project would run on. Docker containers can be deployed anywhere, e.g., virtual machines, cloud, etc. A docker image is a file consisting of instructions to execute the entire application, including the code, the environment variables, libraries, and configuration files. When the docker image is deployed to an environment, it is implemented as a container. A docker image can be instantiated into multiple containers. For this thesis, docker images for various fabric components published in the docker hub are used [46].

6.1.3 React.js

React.js is an open-source JavaScript library used to implement UI components for web apps developed and maintained by Facebook. Some of the distinguishing features are Virtual DOM, Unidirectional data flow and is light-weight. Traditional frameworks were based on DOM for UI rendering. For every change on the page, the DOM needs to re-render the page, which is not ideal for complex applications. The Virtual DOM creates a copy of the previous DOM, which compares with the new DOM to determine the current changes, and only these changes are rendered rather than the entire DOM. The learning curve is also low as developers can get started with basic knowledge of JavaScript, HTML, and CSS.

6.1.4 IPFS

IPFS is a peer-to-peer distributed file sharing system. In a traditional file storage system, addressing a file is IP based, and it can be downloaded from a single server at a time [47], causing the process to consume a lot of bandwidth. The owner of the server can also prevent and restrict access to the file. IPFS does not suffer from this limitations as the files on the IPFS network are addressed using a unique cryptographic hash of the contents. When the user request access to the file, the node closest to him hosting the file will respond. Once the user downloads the file, he becomes a host himself.

6.1.5 Node.js

Node.js is an open-source JavaScript runtime environment build of Chrome’s V8 JS engine [48]. Rather than having multiple languages to implement the front end and backend for a web app, node.js unifies the development into one language. For this thesis, we use Node.js as a backend technology to run the server hosting the APIs that communicate with the hyperledger network.

6.2 Hyperledger Network Description

This section describes the structure of the hyperledger blockchain network implemented in this thesis and is illustrated in Figure 6.1.

The network consists of four organizations called Customer (R1), Manufacturer (R2), Raw Material Supplier (R3) and Component Supplier (R4), and an ordering service. Each organization has its own certificate authority (CA*) to issue the identities to its administrators and nodes and two peers. Peer 1 (P1) hosts the copy of smart contract S1 related to *Transparent Data Flow*, while Peer 2 (P2) hosts the copy of S2 related to *Product Tracking*. For better modularity, two channels are used: Channel 1 contains the ledger (L1) for *Transparent Data Flow* and the P0s of all organizations are attached to it; Channel 2 contains the ledger (L2) for the Product History and all P1s are attached to it.

The Ledger consists of the blockchain and the World State (WS), which is a CouchDB database. L1 holds the list of Order hashes and Order Access requests, L2 holds the list of connections.

For confidential data, the information exchanged between peers communicating through channel 1 is stored in Private Data Collections (PDC). These PDC contain the Orders whose hash is stored in the WS and each peer has 4 PDC, one for itself and three for communication with the other organizations. The total number of shared PDCs is 10.

For example, P0 of Customer contains those four private data collections;

PDC1 - Customer data only
PDC2 - Customer and Manufacturer data
PDC3 - Customer and Raw Material Supplier data
PDC4 - Customer and Component Supplier data.

P0 of Manufacturer contains
PDC5 - Manufacturer data only
PDC6 - Manufacturer and Raw Material Supplier data
PDC7 - Manufacturer and Component Supplier data.
PDC2 - shared with Customer

The client application (A*) can access the smart contracts using P0 and P1 node. It is implemented using Hyperledger Fabric SDK for node.js that provides APIs to communicate with the blockchain network.

The client application is hosted on a node server and provides simplified REST APIs to the front end component. The front end is agnostic to the blockchain network because it is abstracted out on the API side. It is implemented using react js.

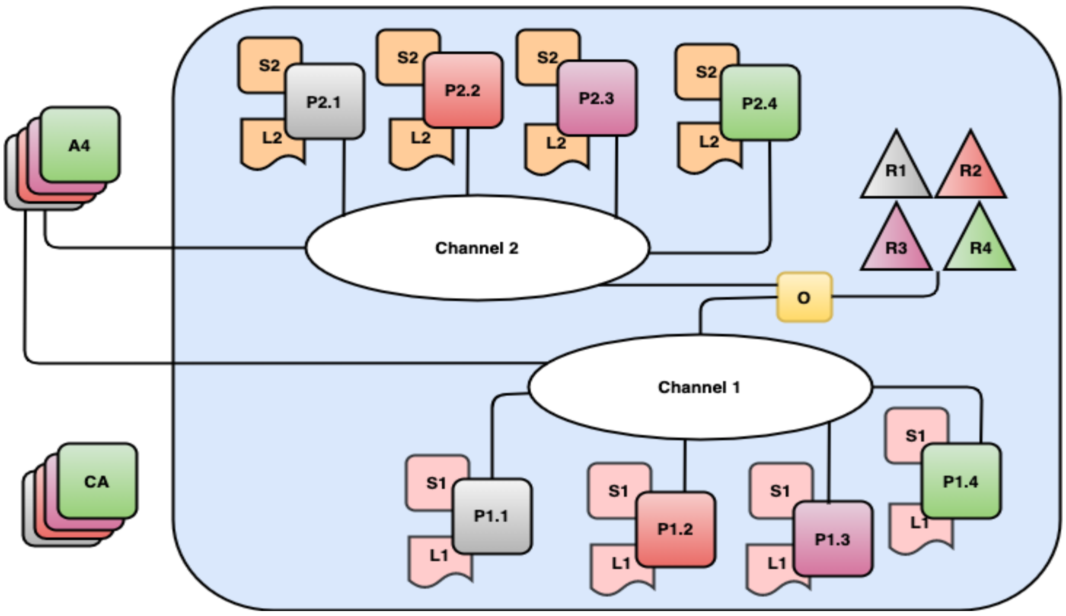


Figure 6.1: Hyperledger Fabric Network for Blockchain-based Supply Chains

6.3 Evaluation based on reference architecture

One of the goals for this thesis is to develop a reference architecture based on the requirements described in chapter 4. The architecture is component-based and is decomposed into logical and functional components. This section describes the requirements and how the architecture fulfills them.

System Requirements

For this thesis, user actions such as creating, updating and deleting the data are recorded as transactions, grouped into blocks and stored in the blockchain. As all nodes in the network have the copy of the blockchain and are synchronized using the consensus mechanism, the blockchain is immutable. Table 6.1 summarizes the evaluation for system requirements with respect to the architecture.

The . . .		should...	Fulfilment
System	R1	Record user actions and store them in the ledger.	Events are recorded as transactions and grouped into blocks and are appended into the blockchain.
	R2	Maintain an immutable list of transactions in the ledger	Consensus mechanism in the blockchain-based system help to maintain an immutable ledger
	R3	Add mechanism to limit access to the ledger	Confidential data is stored in Private Data collections and the hash of the collection is stored in the ledger, which can be later used to validate the authenticity of the the private data by any other party in the network.
	R4	Expose the smart contracts using APIs	The Web backend component provides an API which helps in communicating with the smart contracts.
	R5	Grant authorization users to execute smart contract	The user needs to be authorized before communicating with the smart contracts via the API. This APIs can be only accessed using the authorization token.

Table 6.1: Evaluation of the system architecture based on the system requirements

Transparent Order Details

Each step in the supply chain is stored in the ledger in the form of an order object. Both parties involved in the process need to approve the details stored on the blockchain to avoid data related disputes. The files are stored on a distributed file system, and the hash used to access the file is stored in the order details. The order details are stored in private data collections, and the hash of the data is stored in the ledger. The hash is used for data validation and serves as evidence for audit process. Only authorized parties can have access to the uploaded orders.

As a...	Id	I want/need to ...	Fulfilment
Consumer	R6	Create new order	Files are uploaded in IPFS, the order details are stored in the private data collections, and the hash of the data is stored in the ledger.
	R7	Update order details	The authorized parties can update the order details, and the approval status is set to PENDING.
	R8	Tag the Producer for approval	The Consumer links the Producer to the order details to approve them, at which point the Producer is authorized to view, update, and approve/reject the order.
	R9	Approve/Reject order details changes from Producer	The Consumer approves or rejects the order details only if the Producer updates them.
	R10	View list of all orders	The Consumer accesses the list of all the orders uploaded in the private data collections.
	R11	Restrict access to the order uploaded only to the Producer	The uploaded data is stored in private data collections, and no other organization has access to the uploaded data.
	R12	Restrict access to the order shared only to the authorized organizations	The Consumer can provide authorization to another organization in the network to access the data.
	R13	Delete an order	The data can be deleted from the private data collections, and only the hash of the data remains on the ledger.
	R14	Approve order access request	The Consumer approves/rejects the order access request
	R15	View list of order access requests that need approval	The Consumer accesses the list of all access requests

As a...	Id	I want/need to ...	Fulfilment
Producer	R16	Approve/Reject order details agreement	If the Producer is tagged in the order, it can approve or reject the order details uploaded on the private data collection.
	R17	Update order details	If the Producer is tagged in the order, it can update the order details, and the order's approval status is set to PENDING.
	R18	View list of orders that need approval	The list of all the orders that need approval is presented to the user.
Organization	R19	Raise an access request to view the details of an order.	The Organization, which is not tagged in the order and cannot access the private Order details, raises an access request
	R20	View the list of raised access requests	The Organizations can view the list of access requests that they have raised

Table 6.2: Evaluation of the system architecture based on the Business requirements of the Transparent Product History module

Product Tracking

A traceable product history is retrieved with the help of connections or links stored between any two orders. The product history consists of supply chain process steps with minimum information visible to any user in the network. The data related to the connections is stored on the ledger which can be accessed by all the parties in the network. If the user needs more information regarding a process, they need to raise an access request to view the details.

As an...	Id	I want/need to ...	Fulfilment
Organization	R21	Access the product history using the order id.	Any party in the network can access the product history as a list of processes using the product id. Each step displays the type of process, the organization name, and the date.
	R22	Upload order connections.	The Organization uses a smart contract function to upload the order connection

Table 6.3: Evaluation of the system architecture based on the requirements of the Product Tracking module

6.4 Evaluation based on the prototype implementation

To verify if the designed architecture suits the analyzed requirements, we evaluated it using the prototypical implementation, and the results are summarized below with the requirement, its fulfillment and screenshots.

R1 *The system should record user actions and store them in the ledger.*
All the user actions in the fabric network are stored as transactions in the hyperledger network's ledger. As mentioned in section 6.2, the network has two ledger

R2 *The system should maintain an immutable list of transactions in the ledger.*
Each node in the hyperledger network holds an identical copy of the blockchain achieved by a consensus mechanism. Endorsement policies govern smart contracts' execution and define the list of peers that need to agree on the transaction before adding it to the ledger.

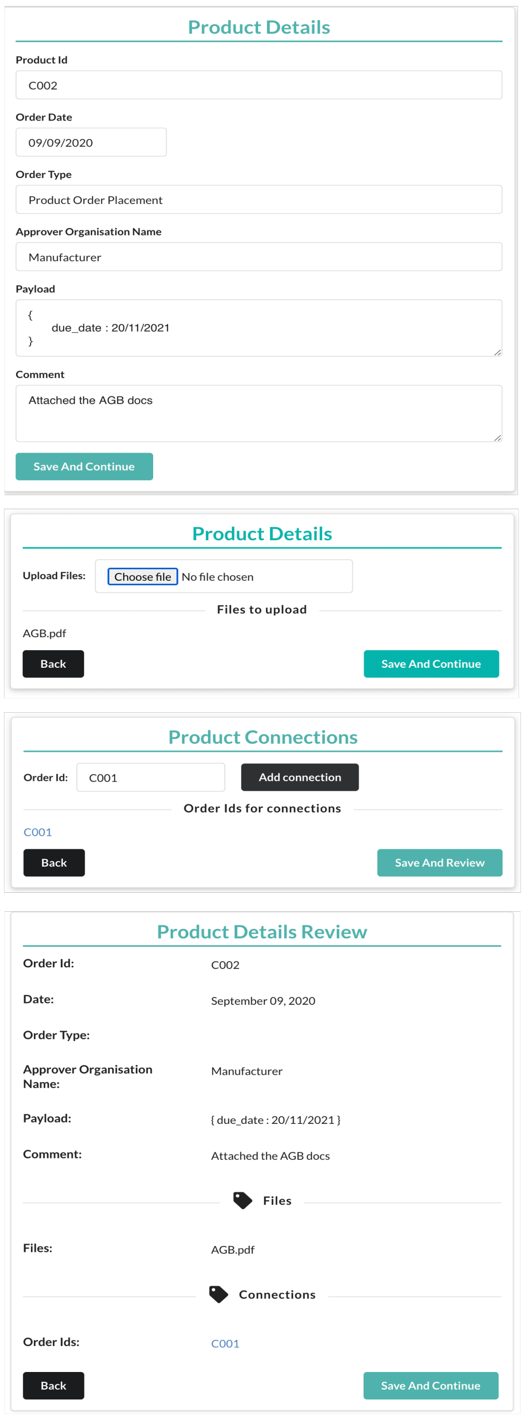
R3 *The system should able to add a mechanism to limit access to the ledger.*
Channels and Private data collections in the fabric network help achieve the confidentiality of private data.

R4 *The system should expose the smart contracts using APIs.*
Hyperledger Fabric SDK includes the fabric-ca-client, which allows applications to communicate with the hyperledger network. We use node.js to implement APIs that interact with the smart contract.

R5 *The system should allow authorized users to execute smart contracts.*

Identity management is implemented using the basic authentication mechanism provided by Fabric Certificate authority. The Membership service provider, a pluggable component provided by Fabric, could be used to abstracts all the cryptographic protocols to provide additional permissions.

R6 *As a Consumer, I want to create an order.*
After logging in, the Customer can navigate to the 'create order page' and enter the details illustrated in Figure 6.2. On successfully creating the order, the Customer is redirected to the 'order details page'. The user can can add text details, upload a file and link another order as connections and in the end review the order before submitting.



Product Details

Product Id: C002

Order Date: 09/09/2020

Order Type: Product Order Placement

Approver Organisation Name: Manufacturer

Payload:
```
{
    due_date : 20/11/2021
}
```

Comment: Attached the AGB docs

Product Details

Upload Files: No file chosen

Files to upload

AGB.pdf

Product Connections

Order Id: C001

Order Ids for connections

C001

Product Details Review

Order Id:	C002
Date:	September 09, 2020
Order Type:	
Approver Organisation Name:	Manufacturer
Payload:	{ due_date : 20/11/2021 }
Comment:	Attached the AGB docs

Files

Files:	AGB.pdf

Connections

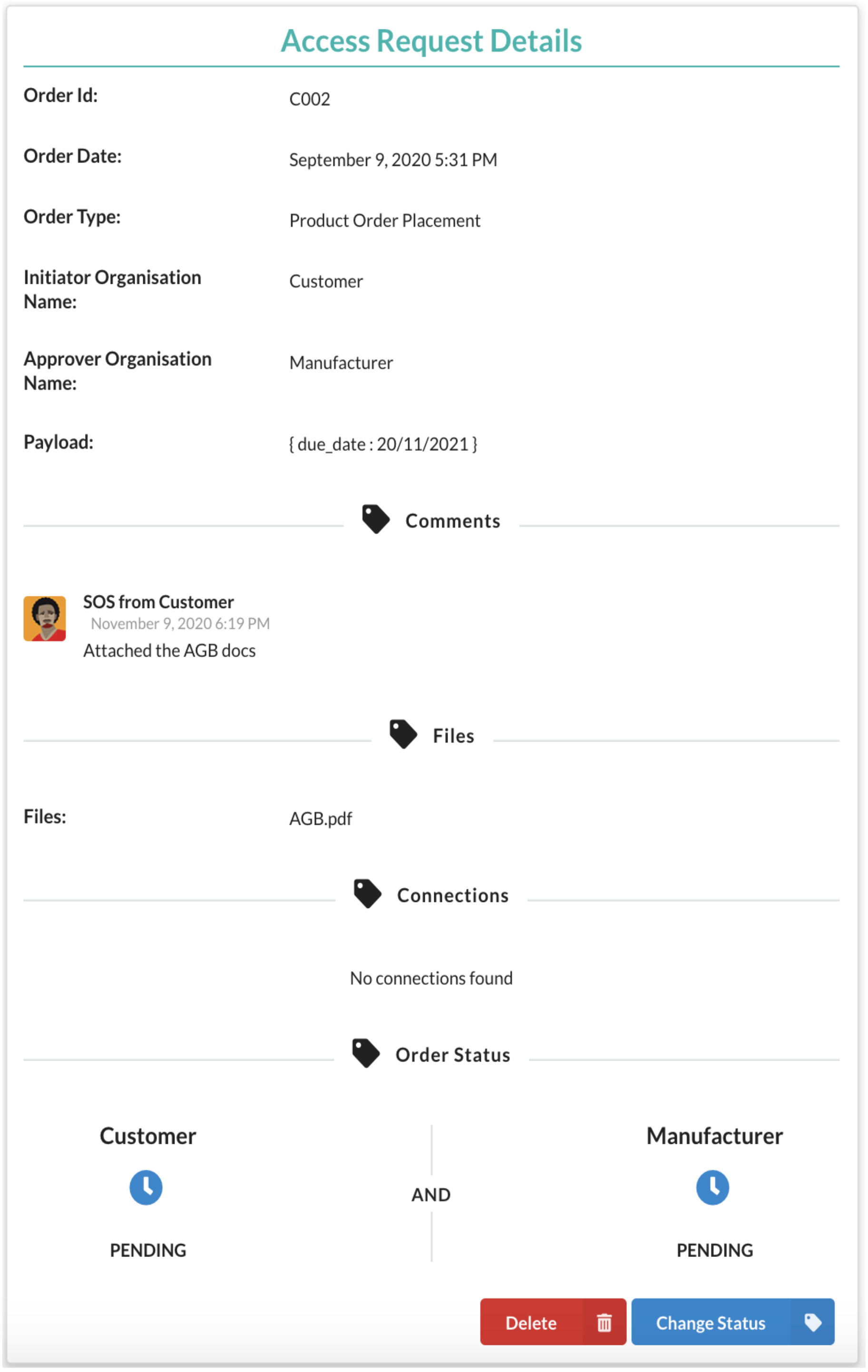
Order Ids:	C001

Figure 6.2: Screenshot for order creation

R7 *As a Consumer, I want to update the order.*
The Customer can click on the update button available on the 'order details page' and update the order details.

R8 *As a Consumer, I want to tag the Producer for approval.*
The Customer can tag a Producer while updating the order using the feature in R7.

R9 *As a Consumer, I want to approve/Reject order details changes from the Producer.*
The Customer can click the button Approve or Reject, and an option to add comments is provided. The button is only visible if the Consumer needs to approve an order. Refer Figure 6.3. The user is registered as Manufacturer and it needs to approve or reject the order by using the 'Change Status' button.


Access Request Details

Order Id:	C002
Order Date:	September 9, 2020 5:31 PM
Order Type:	Product Order Placement
Initiator Organisation Name:	Customer
Approver Organisation Name:	Manufacturer
Payload:	{ due_date : 20/11/2021 }

Comments

SOS from Customer
November 9, 2020 6:19 PM
Attached the AGB docs

Files

Files: AGB.pdf

Connections

No connections found

Order Status

Figure 6.3: Screenshot for order status approval

R10 *As a Consumer, I want to view a list of all orders.*
The Consumer can navigate to view the list of all the orders using the navigation menu.

R11 *As a Consumer, I want to restrict access to the order uploaded only to the Producer.*
An unauthorized Organization cannot access the order details, instead it will receive an error message.

R12 *As a Consumer, I want to restrict access to the order shared only to the authorized organizations.*
Same as R11.

R13 *As a Consumer, I want to delete an order.*
The Consumer can delete the order by clicking the Delete button on the 'view order details' page.

R14 *As a Consumer, I want to approve order access request.*
The private data is copied from one private data collection to the one shared with the requesting Organization. To verify the authenticity of the data, the hash is used for comparison.

R15 *As a Consumer, I want to view list of order access requests that need approval.*
The Consumer can view the list of access requests received.

R16 *As a Producer, I want to approve/Reject the order details agreement.*
Same UI as R9. The button is only visible if the Producer needs to approve the order.

R17 *As a Producer, I want to update the order details.*
Same UI as R7.

R18 *As a Producer, I want to view a list of orders that need approval.*
The Producer can navigate to view the list through the navigation menu.

R19 *As an Organization, I want to raise an access request to view the details of an order.*
The Consumer needs to click the Request Access button and add comments. The Consumer can raise only one request per process step.

R20 *As an Organization, I want to view the list of raised access request.*
The Organization can view the list of raised access requests via the option in the menu.

R21 *As a Consumer, I want to access the product history using the product id.*
The product history can be view as process steps after entering the product id.

R22 *As a Consumer, I want to upload order connections*
Order connections are created when the user adds the another order id while creating the order. In Figure 6.1, illustrates the order creation UI and the page contains a section to add the connections.

7 Conclusion

All the contributions in this thesis aim to solve transparent data flow issues and provide traceable product history with three research questions mentioned in section 1.2.

The integration of blockchain can help improve the supply chain in many factors. The proposed architecture allows various parties in the network to share private data by keeping business secrets confidential. The immutable property of the blockchain makes it easy to verify the authenticity of the data ledger, helping in quickly resolving claims in the case of a dispute between parties. The immediate availability of the data is also a plus compared to standard practices involving slow communication of documents between parties.

The first research question was concluded by conducting a stakeholder analysis, followed by a system and requirement analysis. The system analysis helped design the requirements by analyzing the flaws in the current system. Furthermore, the architecture was designed using Kruchten (1995) "4+1 view model". The prototype was developed using the hyperledger fabric as a blockchain platform by referring the designed architectures. In the last step, the artifacts of the thesis, i.e., the architecture, is evaluated against the requirements using the developed prototype and the results conclude that the architecture satisfies the requirements for a DLT based supply chain system.

7.1 Limitations

This thesis suffered from a methodological and a technical limitation. Methodological limitations concern actions taken during this thesis to account for the limited time and resources. The methodological restriction was due to the focus on IBO's problems, which means the requirements are confined to the manufacturer's side. A more complete and robust solution would be reached by analyzing the other parties' requirements in the supply chain.

From a technical limitation, IPFS was used as a file storage system. IPFS is a distributed peer-to-peer protocol and is based on the concept of content-addressing, i.e., the content uploaded on IPFS can be accessed using its cryptographic hash. The hash of the file is stored on the ledger and can be accessed only by authorized parties. However, any entity in possession of the hash can access the file on the network, rendering uploading of sensitive files unsafe. The only solution is to encrypt the file before uploading it and have decrypting measures for both parties.

7.2 Future Work:

The prototype implemented during this thesis is demonstrative and can be extended in many ways. This includes the following updates:

1. Gathering requirements by interviewing other SMEs and parties in the supply chain.
2. Researching more on the data that needs to be uploaded on the ledger.
3. Implementing encryption and decryption logic for private files uploaded on IPFS.
4. Implementing authorization logic for users in each supply chain organization.

8 Bibliography

[1] *Supply Chain and Logistics Management: Concepts, Methodologies, Tools, and Applications*. IGI Global, 2020. ISBN: 9781799809456. DOI: 10.4018/978-1-7998-0945-6. URL: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-7998-0945-6.

[2] J. Thangaraj and R. Lakshmi Narayanan. "INDUSTRY 1.0 TO 4.0: THE EVOLUTION OF SMART FACTORIES." In: (October 2018).

[3] S. Chopra and P. Meindl. *Supply Chain Management: Strategy, Planning, and Operation*. English (US). 6th. Pearson Education, 2016. ISBN: 97812920093567.

[4] S. S. Pettersen and B. E. Asbjørnslett. "Assessing the Vulnerability of Supply Chains: Advances from Engineering Systems." In: *Revisiting Supply Chain Risk*. Ed. by G. A. Zsidisin and M. Henke. Cham: Springer International Publishing, 2019, pp. 15–35. ISBN: 978-3-030-03813-7. DOI: 10.1007/978-3-030-03813-7_2. URL: https://doi.org/10.1007/978-3-030-03813-7_2.

[5] W. Donald. *Supply Chain Risk Management : Vulnerability and Resilience in Logistics.* Vol. 2nd ed. Kogan Page, 2011, pp. 99–128. ISBN: 9780749463939. URL: http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=395740&site=ehost-live.

[6] P. Yang, J. Tang, and C. Yan. "A Case Study of Global Supply Chain Risk Management." In: *2012 24th Chinese Control and Decision Conference (CCDC)*. 2012, pp. 1996–2000. DOI: 10.1109/CCDC.2012.6243026.

[7] S. GIBBENS. *What is seafood fraud? Dangerous—and running rampant, report finds*. 2019. URL: https://www.nationalgeographic.com/environment/2019/03/study-finds-seafood-mislabeled-illegal/.

[8] K. PEFFERS, T. TUUNANEN, M. A. ROTHENBERGER, and S. CHATTERJEE. "A Design Science Research Methodology for Information Systems Research." In: *Journal of Management Information Systems* 24.3 (2007), pp. 45–77. ISSN: 07421222. URL: http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=28843849&site=ehost-live.

[9] P. Kruchten. “Architecture Blueprints—the “4+1” View Model of Software Architecture.” In: *Tutorial Proceedings on TRI-Ada ’91: Ada’s Role in Global Markets: Solutions for a Changing Complex World*. TRI-Ada ’95. Anaheim, California, USA: Association for Computing Machinery, 1995, pp. 540–555. ISBN: 0897917057. DOI: 10.1145/216591.216611. URL: https://doi.org/10.1145/216591.216611.

[10] W. Kenton. *How General Ledgers Work*. September 2020. URL: https://www.investopedia.com/terms/g/generalledger.asp.

[11] F. M. Benčić and I. Podnar Žarko. “Distributed Ledger Technology: Blockchain Compared to Directed Acyclic Graph.” In: *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. 2018, pp. 1569–1570. DOI: 10.1109/ICDCS.2018.00171.

[12] K. Sultan, U. Ruhi, and R. Lakhani. *Conceptualizing Blockchains: Characteristics & Applications*. 2018. arXiv: 1806.03693 [cs.CY].

[13] K. Raj. *Foundations of Blockchain*. 2019. URL: https://learning.oreilly.com/library/view/foundations-of-blockchain/9781789139396/.

[14] N. Subramanian, A. Chaudhuri, and Y. Kayikci. “Basics of Blockchain.” In: *Blockchain and Supply Chain Logistics: Evolutionary Case Studies*. Cham: Springer International Publishing, 2020, pp. 11–19. ISBN: 978-3-030-47531-4. DOI: 10.1007/978-3-030-47531-4_2. URL: https://doi.org/10.1007/978-3-030-47531-4_2.

[15] B. Academy. *Private, Public, and Consortium Blockchains - What’s the Difference?* October 2020. URL: https://academy.binance.com/en/articles/private-public-and-consortium-blockchains-whats-the-difference.

[16] S. Nakamoto. *Bitcoin: A peer-to-peer electronic cash system,” http://bitcoin.org/bitcoin.pdf*.

[17] *What is Ethereum?* URL: https://ethereum.org/en/what-is-ethereum/.

[18] URL: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html.

[19] *Learn about Ethereum*. URL: https://ethereum.org/en/learn/#smart-contracts.

[20] *Enterprise Blockchain Platform: Corda Platform and Services by R3*. URL: https://www.r3.com/corda-platform/.

[21] R. Brown, J. Carlyle, I. Grigg, and M. Hearn. *Corda: An Introduction*. September 2016. DOI: 10.13140/RG.2.2.30487.37284.

[22] *Introduction¶*. URL: https://hyperledger-fabric.readthedocs.io/en/release-2.2/whatis.html.

[23] *Hyperledger Fabric Model¶*. URL: https://hyperledger-fabric.readthedocs.io/en/release-2.2/fabric_model.html.

[24] *The Ordering Service¶*. URL: https://hyperledger-fabric.readthedocs.io/en/release-2.2/orderer/ordering_service.html.

[25] *Identity¶*. URL: https://hyperledger-fabric.readthedocs.io/en/release-2.2/identity/identity.html.

[26] *Peers¶*. URL: https://hyperledger-fabric.readthedocs.io/en/release-2.2/peers/peers.html.

[27] *Private data¶*. URL: https://hyperledger-fabric.readthedocs.io/en/release-2.2/private-data/private-data.html.

[28] *Blockchain network¶*. URL: https://hyperledger-fabric.readthedocs.io/en/release-2.2/network/network.html.

[29] *What is IPFS?* September 2020. URL: https://docs.ipfs.io/concepts/what-is-ipfs/.

[30] *Persistence*. October 2020. URL: https://docs.ipfs.io/concepts/persistence/.

[31] *Diamonds*. October 2020. URL: https://www.everledger.io/industry-solutions/diamonds/.

[32] *OriginTrail. Making Supply Chains Work. Together.* URL: https://tech.origintrail.io/protocol.

[33] URL: https://www.ibo-tec.de/de/produkte/alle-produkte.html.

[34] URL: https://www.ibo-tec.de/de/service/unser-service.html.

[35] K. Peffers, T. Tuunanen, M. Rothenberger, and S. Chatterjee. "A Design Science Research Methodology for Information Systems Research." In: 24.3 (December 2007), pp. 45–77. ISSN: 0742-1222. DOI: 10.2753/MIS0742-1222240302. URL: https://doi.org/10.2753/MIS0742-1222240302.

[36] R. Freeman and J. Mcvea. "A Stakeholder Approach to Strategic Management." In: *SSRN Electronic Journal* (January 2001). DOI: 10.2139/ssrn.263511.

[37] CSCMP, H. Chen, C. Defee, and B. J. Gibson. *Defining the Supply Chain*. January 2014. URL: https://www.informit.com/articles/article.aspx?p=2166717&seqNum=3.

[38] D. Silver. *The Automotive Supply Chain, Explained*. May 2016. URL: https://medium.com/self-driving-cars/the-automotive-supply-chain-explained-d4e74250106f.

[39] D. Balta, V. Greger, P. Wolf, and H. Krcmar. "E-government Stakeholder Analysis and Management Based on Stakeholder Interactions and Resource Dependencies." In: *2015 48th Hawaii International Conference on System Sciences*. 2015, pp. 2456–2465. DOI: 10.1109/HICSS.2015.294.

[40] *Fines / Penalties*. July 2020. URL: https://gdpr-info.eu/issues/fines-penalties/#:~:text=83(4)%20GDPR%20sets%20forth,to%20that%20used%20in%20Art.

[41] URL: https://www.europarl.europa.eu/thinktank/en/document.html?reference=EPRS_STU(2019)634445.

[42] G. Alter, Browne, S. Alter, and G. Browne. "A Broad View of Systems Analysis and Design: Implications for Research." In: *Communications of the Association for Information Systems* 15 (January 2005), pp. 981–999. DOI: 10.17705/1CAIS.01650.

[43] D. Garlan and D. Perry. "Introduction to the Special Issue on Software Architecture." In: *IEEE Transactions on Software Engineering* 21.4 (April 1995).

[44] S. Yongchareon, C. Liu, X. Zhao, and M. Kowalkiewicz. "BPMN Process Views Construction." In: vol. 5981. April 2010, pp. 550–564. DOI: 10.1007/978-3-642-12026-8_42.

[45] *Open Source Blockchain Technologies*. October 2020. URL: https://www.hyperledger.org/.

[46] URL: https://hub.docker.com/u/hyperledger/.

[47] J. Bostoen. *A Hands-on Introduction to IPFS*. October 2020. URL: https://medium.com/coinmonks/a-hands-on-introduction-to-ipfs-ee65b594937.

[48] Node.js. *About Node.js*. URL: https://nodejs.org/en/about/.