UNIVERSITY OF PORTO • FACULTY OF ENGINEERING

Supply Chain Management with Blockchain Technologies

Pedro Miguel Lourenço Costa

July 2018

Scientific Supervision by

Filipe Figueiredo Correia, Assistant Professor
Department of Informatics Engineering

In partial fulfillment of requirements for the degree of
Master in Informatics and Computing Engineering
by the EUR-ACE Programme

Contact Information:

Pedro Miguel Lourenço Costa
Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Informática

Rua Dr. Roberto Frias, s/n
4200-465 Porto
Portugal

Tel.: +351 968 203 386
Email: pedro.miguel.costa@fe.up.pt
URL: https://www.linkedin.com/in/pedro-costa-3279996b

This page was intentionally left blank.

Abstract

Supply chain refers to the flow of products and associated information which are exchanged between companies. The direction of this flow goes from supplier to consumer, with complex exchanges and transformations happening between the origin and delivery of a product. Thus, supply chain management (SCM) is essential in the coordination of a supply chain.

A supply chain faces many difficulties: traceability and provenance of the products, inventory management, quality control and schedules to follow are some. Delays are common, affecting a company's finances, growth and reputation. This is aggravated by the fact that the needed information is not always accurate or available, a consequence, in part, of the manual processes for inserting information into the companies' systems and of the non-existence of reliable technologies which can integrate the information in a secure and fast manner.

One way to approach these problems of supply chains is to prioritize them and update the supporting technologies. A seemingly adequate technology is the blockchain distributed architecture, which allows for the immutable and secure storage of data. This might be a good way to achieve traceability of a supply chain, possibly leading to turning the chain fully digital and automated.

This dissertation focuses on researching the extent of the issues of supply chains, and whether blockchain can be used to solve these issues. Therefore, the hypothesis is that blockchain architectures can be a good fit for supply chain management.

To validate the issues of SCM and elicit requirements for a supply chain software system, a survey was conducted. By using these requirements, it was possible to propose and iteratively build a blockchain architectural solution, aiming to test whether the gathered requirements are feasible to be implemented.

In the end, the solution was analyzed against the elicited requirements, in order to verify the degree to which it was possible to implement them. From this verification, it was possible to draw some conclusions. The fact is that the designed architecture could prove to handle most of the requirements for a supply chain, but the implementation

of other requirements remains to be achieved. The expectation is that the contributions, both of the successes and failures, might help pave the way for future work and research in integrating blockchain architecture into SCM software systems.

Keywords: Supply Chain, Supply Chain Management, Requirements Engineering, Blockchain, Distributed Architecture, Security, Traceability, Information Integration, Interoperability

Resumo

Empresas de qualquer indústria trocam informação e recursos entre elas, criando um fluxo comummente chamado de cadeia de logística. A direção deste fluxo vai do fornecedor até ao consumidor, com trocas e transformações complexas a acontecer desde a origem até à entrega final do produto. Assim, a gestão da cadeia de logística é essential para coordenar esta cadeia.

A cadeia de logística enfrenta muitas dificuldades: rastreabilidade de produtos, gestão de inventário, controlo de qualidade e de prazos são apenas algumas. Atrasos são comuns e podem afetar as finanças, crescimentoe reputação de uma empresa. A adicionar a estas dificuldades, muitas vezes a informação necessária a certos processos não está disponível ou não é exata, consequencia, em parte, dos processos manuais usados para inserir informação nos sistemas, o que é lento e pode levar a erros. Isto poderá ser causado pela não existencia de tecnologias de confiança que possam integrar toda a informação de forma segura e rápida.

Uma forma de abordar estes problemas da cadeia de logística é priorizá-los e atualizar as tecnologias de suporte da cadeia de logística. Uma tecnologia que parece adequada é a arquitetura distribuída de blockchain. Esta é uma tecnologia distribuida que permite o armazenamento seguro e imutável de dados, levando a que a informação esteja acessível em qualquer lugar, a qualquer hora. Assim, as blockchains podem ser o meio mais adequado de atingir a rastreabilidadede uma cadeia de logística, possivelmente levando a que esta se torne completamente digital.

Esta dissertação foca-se em pesquisar os problemas da gestão de cadeias de logística e a que nível poderão ser aplicadas blockchains de forma benéfica para resolver esses problemas. A hipótese é que as blockchain possam ser uma arquitetura adequada para sistemas de gestão das cadeias de logística.

Para validar estes problemas e levantar requisitos para um sistema de software de logística, foi realizado um inquérito. Usando estes requisitos, foi possível propor e iterativamente construir uma arquitetura de blockchain, para testar se os requisitos eram possíveis de ser implementados.

No final, o sistema proposto foi analisado e comparado com os requisitos, para verificar se a implementação foi possível. Daqui foi possível tirar algumas conclusões, nomeadamente que esta arquitetura provou conseguir cumprir a maior parte dos requisitos, mas a implementação de outros não foi conseguida e permanece por provar que consiga ser feita. Assim, espera-se que as contribuições do trabalho aqui realizado possam ser uma base para trabalhos futuros de investigação em integrar esta arquitetura com sistemas reais de gestão de cadeias de logística, nomeadamente a partir dos requisitos implementados.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Our society is turning more consumerist, with most people in developed countries having high consumer power and standards of life. Consumers goods, from the essentials up until the entertainment goods, are manufactured and ordered continuously in high quantities.

Modern supply chains feel the pressure of this growth, leading to the demand for an efficient management. Most companies are making efforts towards this end, and, even though part of the answer to an efficient management lies in having efficient processes, a good management is also based in using the right technologies. Thus, the development of technologies that can satisfy the demands of supply chain management, for any industry, is in high demand.

1.1 Supply Chains

Supply Chains can be found, in some form, in nearly every business, spanning many different areas of operation. Traditionally, a supply chain encompasses all the processes and activities that lead from the initial raw materials to the final finished product, as well as all the functions and services within and outside a company. A supply chain can also be defined as the network of entities through which material flows. These entities can be identified as suppliers, carriers, manufacturing sites, distribution centers, retailers, and customers [1]. Naturally, with the upstream and downstream flow of these materials and resources, comes a lot of information on them and on the processes, people and organizations they are associated with. Realistically, the flow is not always arborescent, as there are many considerations to be taken and decisions to be made. Supply chains have multiple end products with shared components, facilities

and capacities [2]. As a consequence, the paths taken by the resources and information are not straightforward, but interlace, diverge and converge at different points, go back and forth, as exemplified in Figure 1.1.

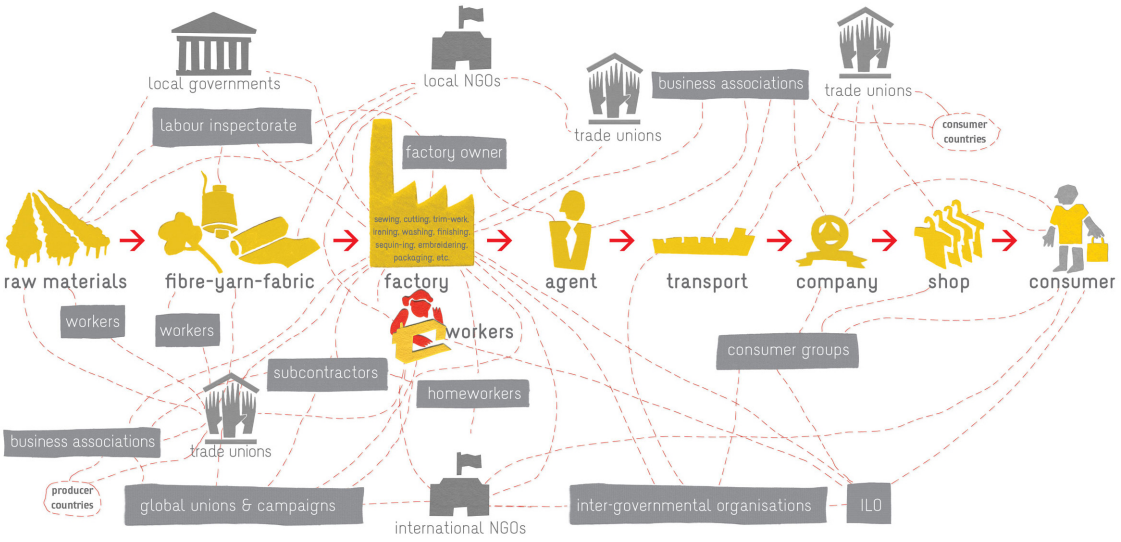


Figure 1.1: Representation of a garment supply chain and all the relationships it involves. Taken from the International Training Centre of the International Labour Organization briefing on global supply chains [3].

The activities and processes a supply chain encompasses include: sourcing raw materials and parts, manufacturing and assembly, warehousing and inventory tracking, order entry and order management, distribution across all channels, delivery to the customer, and managing the information systems necessary to monitor all of these activities. As Lummus [1] describes, these activities can be roughly mapped to the 4 essential processes: plan, source, make, deliver.

Coordinating all of these is no easy task, and so the discipline of SCM comes into life. According to Ballou [4], the Council of SCM Professionals (CSCMP) defines SCM as *"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. In essence, SCM integrates supply and demand management within and across companies"*.

From this definition follows that SCM deals a lot with both coordination and collaboration between entities, and so, the management of the flow of information and resources between them is very important. The objective is always, of course, to minimize the total cost of these flows between and among stages [5]. In the end, the creation of value (products and services) in a supply chain stems from the relationships

that different entities build between themselves, and not from the work of a single entity. As such, supply chains, not firms, compete and those which have the best integration and management processes win.

And this is where SCM shines and shows just how useful it can be. Managing all the processes in a supply chain, while maintaining safety, quality and keeping to schedule is difficult. An event on one side of the world, large or small, be it from human or natural causes, can easily disrupt the links in the supply chain. For instance, it might disrupt the supply of a critical component or service. Delays are, therefore, common, and the consequences of such disruptions might have a severe impact in the finances, growth and reputation of the companies involved [6].

SCM diminishes the impact of such disruptions, and actively works to avoid or diminish them, while optimizing the way the supply chain works. This is why SCM is such an important discipline, that we have to better understand and improve, with all the means that we can, and this includes, of course, research into technologies like the blockchain.

1.2 Challenges

Having already introduced the concepts of Supply Chain and SCM, it is now possible to briefly introduce some of the problems that affect them.

The first, and most generalist problem of a supply chain, is the ease with which **an unexpected event might cause delays**. These events, already mentioned in Section 1.1 are not always predictable and must be contained as fast as possible. One event in particular which, often, causes delays are **synchronization problems in the processes and information systems of a company** [7].

Another problem is that, often, there are difficulties in sharing information between companies. This is caused both by the fact that **companies value their privacy and the security of their information**, which means they might not want to share too much information, or that they might only share it through secure channels, and by the **lack of standards for sending information and communicating** [8]. The issue with non-existing standards is that companies are left to discuss what details to share or not, wasting time and resources.

Most important of all, in the industry, **the use of traditional tools and manual work is still too prevalent**. Emails are sent, documents are printed and mailed, instead of transmitting the information in a more automatic, direct and secure way through the

network. This point also brings the next problem of supply chains to light: the apparent lack of interoperability between certain softwares (which might be a byproduct of by the lack of standards).

Finally, provenance and traceability of the products on a supply chain are a big objective for companies. But **the current technologies used in supply chain only accomplish provenance and traceability in a limited scope**, as the information a certain entity possesses is usually also limited. And so, it is very hard for anyone to have a global overview of the supply chain.

Though it is not proven, it is be possible that some, if not most, of these issues in supply chain might be caused by the use of software architectures that do not allow for full data integration. An optimal supply chain should be as efficient and effective as possible, while being secure and satisfying all the traceability requirements. Perhaps, it is time to try out new solutions which replace or augment the existing ones, in such a way that supply chain management can better satisfy the needed requirements.

1.3 From Blockchain Technologies to Supply Chains

Blockchain technology allows for secure, public, distributed and decentralized systems. Though it was first proposed in its actual form by Satoshi Nakamoto [9], an anonymous group which published a white paper in 2008, this was not the first reference to such a technology. The first work on a cryptographically secured chain of blocks was described in 1991 by Stuart Haber and W. Scott Stornetta [10], and further refined in 1992 by Bayer & Haber, by incorporating Merkle trees [11]. Since then, it has come a long way, sprouting multiple different uses and applications of the technology. Its characteristics make the development of distributed and permanently, globally available systems possible, which is a paradigm that is attracting the interest of various industries.

One area in particular where we believe blockchain could bring about great improvements is Supply Chain Management (SCM). SCM has seen an increase in complexity in the last few decades, due to the globalization of the market, with businesses interlacing in many different ways, their relations extending way beyond what they used to, as found by Filiz Isik [12]. This increase in complexity is somewhat hard to manage and some supply chains stretch and encompass so many businesses that, due to their software not being prepared for this, the information is not always transmitted from end to end, leaving holes of information in between the links that join each business,

thus leading to a lot of chaos and uncertainty as to the state of the key items in the chain [13].

This dissertation work will focus on supply chain management, and on how blockchains can possibly be applied to improve this area, leading to positive impacts in the logistics industry and eventually finding benefits for the consumer as well.

1.4 Motivation

As described in Section 1.2, a possible cause of these problems might be the use of software that can not keep up with the evolution of the requirements of modern supply chains. Today's supply chains have high standards for their requirements and even when the software works just fine, maybe it is not recent enough or it was not specified and built to satisfy these requirements. For instance, product falsification might be a huge issue nowadays in the supply chain, but maybe it was not rated as a high importance problem 15 years ago. Therefore, the software from 15 years ago complied with different requirements than the ones from today and was not built to handle that specific problem well.

Requirements evolve, and so should the technology, in order to support them. There is an immediate need for solutions, which might either completely replace the previous ones, or augment them.

One way to approach these specific problems is to research what would a modern requirements specification for supply chain look like, and develop new technologies that are more accurately specified for the supply problem challenges in question.

The characteristics of blockchain architectures seem to be a good solution for many of the identified problems in supply chain to be reduced or neutralized. These architectures are the perfect means to achieve traceability of a supply chain, and so, they are good to achieve provenance as well. At the same time they are a secure, incorruptible and immutable way to store information, with a fast synchronization time, being perpetually available to anyone who has permission, anywhere within the network. It would also be the way to close the analog gaps, turning the chain fully digital, leading to the possibility of a global overview.

1.5 Objectives

The main objective of this dissertation is to find whether blockchain technology is a good fit to solve the most common problems of supply chain management, and also to find out the technological requirements of a modern supply chain. There are a multitude of small tasks that blockchain could automatize in supply chain, so this thesis will try to figure out which ones blockchain applies to better.

The primary objective of this dissertation is determining if blockchain architectures can be successfully applied to supply chain management as an improvement towards the technologies that are already being used. For this purpose, it is necessary to conduct research on the supply chain issues and validate these, in order to propose a blockchain design that can target these issues successfully.

1.6 Dissertation Structure

Besides the introduction, this dissertation has 6 more chapters, divided into 2 parts.

Part 1: Background and State of the Art - Provides the needed concepts to understanding the work, as well as explanations about the existing tools and projects related to the blockchain and to its application to supply chain management domain. The state of the art is divided into 2 Chapters: 2 and 3.

- Chapter 2, "Blockchain Technologies", some important concepts from the blockchain architecture are presented and discussed, followed by an analysis of the existing blockchain networks and frameworks and their comparison.

- Chapter 3, "Blockchain in the Industry", the application of blockchain to supply chain is discussed in more detail, including the possible advantages, challenges, and following it up with an analysis of the existing blockchain applications that also focus on supply chain.

Part 2: Problem, Research and Solution - Provides a thorough explanation of the problem, objectives and the proposed methodology to approach them. Follows up with a survey and requirements analysis, which serves as a base to propose a blockchain solution design.

- Chapter 4, "Problem Statement", specifies the thesis statement and the questions that need to be answered in order to reach a conclusion.

- Chapter 5, "Supply Chain Issues Validation and Requirements Elicitation", features a survey and the analysis of its results, such as to gather the requirements for a supply chain system, so that a blockchain-based solution can be proposed.
- Chapter 6, "Solution Design and Implementation", features the choice of a blockchain framework and the proposal of a solution, which consists in building a proof of concept (PoC) project that implements the requirements elicited from the survey.

Part 3: Conclusion - Gathers all the information from the results of the solution to make a statement, also listing contributions, difficulties and future work.

- Chapter 7, "Conclusions", features an overview of the work done in the dissertation, providing an answer to the previously defined problem, and following it up with possibilities of future work in this area.

Chapter 2

Blockchain Technologies

This chapter introduces the most important concepts of blockchain which are essential for its applications. It interleaves the features of blockchain with its applicability to supply chain, by highlighting both the disadvantages and disadvantages, of blockchain in general and in particular to SCM, as well as any existing models for the integration of blockchain with SCM.

2.1 Introduction

"The Byzantine Generals' Problem" is a classical problem faced by distributed systems, which, in simple terms, states that consistency in a distributed system can never be fully guaranteed [14]. This is derived from a lack of general consensus as to what the state of the system is at any given time.

Nakamoto's implementation of a blockchain seems to be, at present, the most practical way to approach this problem (though with its limitations), even earning the term "Practical Byzantine Fault Tolerance (PBFT) Algorithm", because of how it handles the consensus issue. The fact that many kinds of applications rely on distributed architectures, which might face these problems, turns blockchain all the more appealing.

Some improvements have been built upon the traditional blockchain, such as smart contracts, which further enhance its use and allow for a wider variety of applications. Some areas where blockchain is starting to see some use include insurance, finance, Internet-of-Things, health care, identity management, to name a few.

In each of these areas, there are many ways in which blockchain can be used, be it to store information, process information, process payments or provide services. The

versatility is what makes it so popular and what allows for the possibility of many other applications to be proposed.

2.2 Core Concepts and Features

The original blockchain, Bitcoin, was developed with the purpose of creating a distributed online payment system without the need for a financial institution or any other centralized trusted third party to verify the transactions. It even has its own currency, made up from digital tokens that represent real money, a concept which is now commonly known as a cryptocurrency.

Eventually, this concept evolved into something more, and new uses, other than online transactions, emerged from the blockchain technology. As described by Marc Pilkington [15], *"the essence of the blockchain is informational before being economic or monetary, conducive to many emerging and increasingly popular token-free blockchains"*.

Therefore, the algorithms used by Bitcoin are not a defining feature of the blockchain technology, but merely one of the many possible applications.

The blockchain itself consists of a peer-to-peer network which continually stores and updates a chronological chain of blocks, where each block stores whatever information is relevant to the system in question, as well as the hash of the previous block. In the original blockchain, each block stored information about transactions between people, where there was a set of rules to check whether the transactions were valid or not. We will get to how a blockchain can be built in a moment.

One important and defining property of the blockchain is that the hash of the previous block also serves as an address. Therefore, each block has a pointer to the previous block, which is known as an hash pointer. This concept is illustrated in Figure 2.1.

2.2.1 Immutability

This hash thus guarantees that the blocks are linked to each other, but it also guarantees the integrity of the block chain. If one block is altered, the hash on the block that follows it stops matching the block's. If we really wanted to change something on this first block, we would have to alter the hash on the following block to match. But then, that second block would also have been altered, and we would have to change the hash on the third block to match, and so on. This means that the blockchain is immutable, as it is not possible to alter a single block without manually altering all the others.

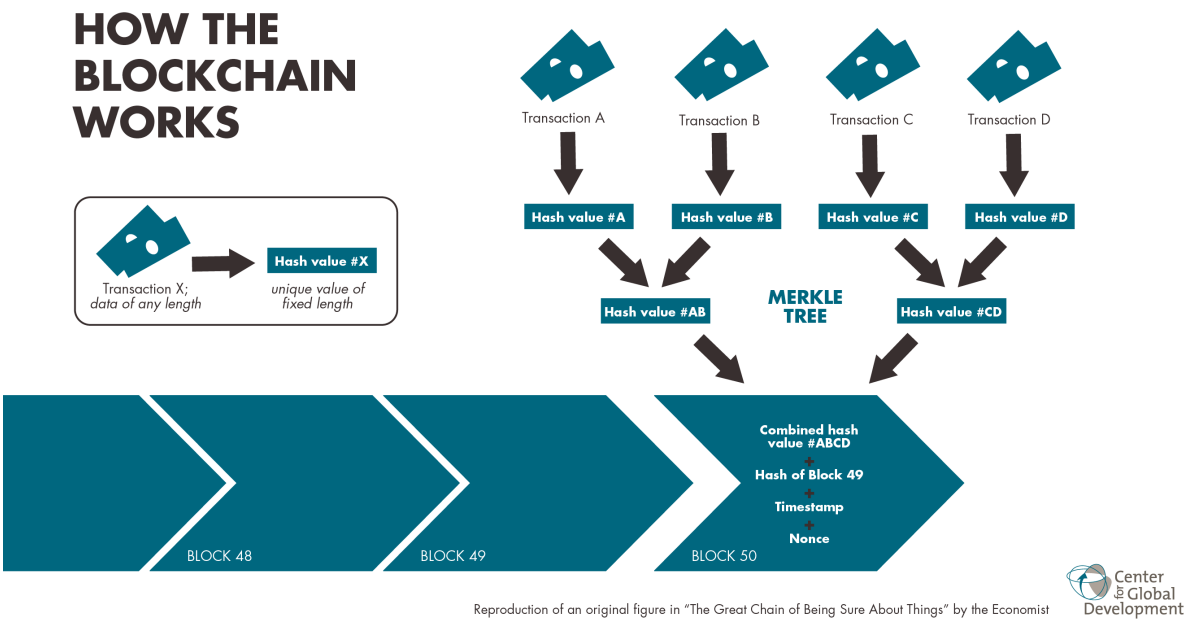


Figure 2.1: Representation of a blockchain's structure

2.2.2 Consensus

The blockchain and its data exists only in a **peer-to-peer network**, and as such, it is stored and extended by the nodes of the network, which form a topology between them. The nodes are machines that have the core code of the blockchain system and which receive and share among themselves the incoming information, in the form of blocks, validating it according to the established rule set. All the nodes, if they are not malicious and actively attempting to change the contents of the chain, contain the same blockchain structure and information, as they all agree on its contents through a consensus algorithm.

Some nodes are open to the internet and the world, thus receiving information from outside the peer-to-peer network and disseminating it to the rest of the network's nodes. A subset of the nodes, called miners, will then gather the circulating information from the peer-to-peer network (by receiving transactions from the nodes they are connected to), and form blocks of information, which they try adding to the blockchain. Obviously, it is impossible for all the nodes to add their blocks at the same time, as each of them would then have a different version of the blockchain and they would get desynchronized. And so, the nodes must reach a consensus as to which block gets added next to the blockchain. For this very reason, the code of the blockchain must have both a consensus algorithm and also a block validation algorithm.

In public blockchains, the mining nodes do all the hard work, and they usually

need some incentive to do so. Cryptocurrencies are virtual currencies that only exist on the blockchain they belong to, and they allow for a new monetary system to come into life. These currencies play an important role in blockchain, since they allow for miners to be rewarded. Without them, public blockchains would probably not work as well, which is why there are so many alternative currencies popping up, with the advent of this technology.

Of course, this is just one of many ways to make a blockchain move forward successfully, and other types of consensus algorithms have been idealized, though most of the consensus algorithms in public blockchains will use cryptocurrencies as the prime incentive.

2.2.3 Private, Public and Hybrid Blockchains

Blockchain, traditionally, is of public nature. But, as many institutions grew aware of the possible benefits of this technology, they started investing in it, and so appeared the private blockchains and the semi-private or consortium blockchains.

Public Blockchains When a blockchain is public, it is accessible to any user. Anyone is able to both read and validate information from it, as well as contribute to its extension by participating in the consensus process. They are secured by the economic incentives that are given to the miners, in the form of cryptocurrency tokens. These blockchains can be considered fully decentralized and have no access control. Every user is at the same level.

Private Blockchains These are usually owned by some organization and have access control, restricting write access to certain peers inside the organization. Read access might be restricted or not, according to the organization's goals. These blockchains, unlike public ones, might not require cryptocurrency or incentives, as the maintenance of the blockchain is done by the organization who owns it, and so they have all the interest in having nodes that can run the consensus process. In the end, this is a more centralized version of the blockchain, as the nodes are concentrated under the ownership of the organization. This also gives the advantage that alterations to the blockchain are easier to achieve, if so is desired (though it subtracts from the actual meaning and concept of an immutable blockchain). It is usually more efficient than public blockchains, being able of achieving a higher number of transactions processed per second.

Consortium blockchains They are a kind of hybrid blockchain, though closer to a private blockchain. They are controlled by many different organizations, and the consensus process is handled by pre-selected nodes from the organizations. This consensus might involve, for example, at least a certain percentage of the nodes agreeing on something (e.g.: if there are 15 fixed nodes, require at least 10 to sign the block). The permissions to read might be public or otherwise permissioned, and as such, this can also be considered a partially decentralized blockchain.

Consortium and private blockchains are not that different. According to Vitalik Buterin, [16], *"so far there has been little emphasis on the distinction between consortium blockchains and fully private blockchains, although it is important: the former provides a hybrid between the 'low-trust' provided by public blockchains and the 'single highly-trusted entity' model of private blockchains, whereas the latter can be more accurately described as a traditional centralized system with a degree of cryptographic auditability attached"*.

2.2.4 Types of Consensus Mechanisms and Algorithms

The most commonly used consensus algorithm is **Proof of Work (PoW)**, though there are others, like the most used alternatives, **Proof of Stake (PoS)**, and **Proof of Activity (PoA)**. We showcase some of the algorithms in Table 2.2.

2.2.5 Transparency

The blockchain is not only available to the mining nodes in the network. Though they are the ones who actively interact with the chain in order to make it grow, if the chain is public, anyone can view the records and verify the authenticity of the data. This is the property of transparency. In private or permissioned blockchains, it might happen that only certain actors have access to certain records, according to the access control rules set by the organization or set of organizations that manage it. This is done with the help of sets of asymmetric key pairs and a public key infrastructure.

Table 2.1: Comparison between different consensus mechanisms.

Type	Overview	Better Used In
Proof of Work (PoW)	In short, PoW works by making the nodes spend computational power until they can find out a hash that satisfies a certain rule, for a certain block. When a node finds this hash, it is allowed to extend the blockchain with that block. The node transmits the new blockchain to all the other nodes. It is assumed that the longed valid chain (where all blocks have valid mining hashes and valid contents) held by any block is the correct one. Additionally, the creator of a block includes a reward for themselves in the block.	Public Blockchains (Bitcoin, alt coins)
Proof of Stake (PoS)	In PoS, the "miners" stake their cryptocurrency tokens as a bet, on which block they want to include in the blockchain. By doing so, they actively have a chance to mint that block proportional to the number of tokens they stake. This makes it so that any participant of the network has in its best interest to be honest. The higher their stake, the more invested in the network they are. PoS is less wasteful than PoW, which consumes a lot of energy in computational power.	Public Blockchains, Consortium Blockchains
Proof of Authority (PoA)	The transactions are validated, aggregated into blocks and put into the blockchain by approved known nodes, which act like "admins" and are the source of truth for the system. This is a more centralized kind of consensus.	Private Blockchains
Proof of Elapsed Time (PoET)	Every participant in the network is assigned a random amount of time to wait, and the first participant to finish waiting gets to commit the next block.	Private Blockchains, Consortium Blockchains
Byzantine Fault Tolerance (BFT)	There are many algorithms for this kind of consensus. One is Practical BFT (PBFT), with pre-selected nodes selecting and ordering the transactions.	Private Blockchains, Consortium Blockchains

2.3 Applications

In general, blockchains are applied in different ways, from public ledgers to private, in a continuum, according to the specific needs. Many of them even apply the concept of smart contracts. Vitalik himself suggested some possible applications of Ethereum, and many more have been idealized, with some even being successfully applied. Here are some examples of areas where the benefits of introducing blockchain have been studied.

- Identity/Record Management - like in a notary, documents are validated and recorded.
- Insurance - Smart contracts, having to abide to certain rule with certainty, make for a perfect system for risk-management. Given that certain conditions are met, the insurance could be claimed, giving way to faster and less error-prone insurance processing.
- Health care - Health records easily accessible anywhere. This could be coupled with other applications, like using sensors and smart contracts to automatically monitor patient status.
- Distributed Cloud Storage - Instead of the traditional centralized cloud, distributed clouds could become a reality.
- Voting - Using blockchain, digital voting could become feasible. The greatest barrier to e-voting have been the concerns with security, and blockchain provides an anonymous and secure way to do it.
- Internet-of-Things (IoT) - Any object connected to the Internet can upload information, which can either be stored or processed on the blockchain. Devices with sensors, for instance, can be programmed to send their values to the blockchain, which can then be queried by others to check these values. There can even be pre-programmed smart contracts that have events based on what is happening and the information that is being sent by the devices.

2.4 Blockchain Frameworks

There are several frameworks to build blockchains, as well as platforms to participate in already existing ones. A blockchain does not have to be built from scratch, as there is already a lot of work done in this area. This chapter goes into detail, explaining some of the most important frameworks, and their applicability to certain cases.

2.4.1 Ethereum

With the purpose of building a blockchain that does more than just provide a currency system, the Ethereum project was developed and launched in 2014. A white paper, written by Vitalik Buterin was released explaining the concepts and inner-workings of this platform, and its popularity has been growing ever since. Citing Buterin's paper, Ethereum has the intent to *"allow developers to create arbitrary consensus-based applications that have the scalability, standardization, feature-completeness, ease of development and interoperability"* [17].

Smart Contracts

In short, Ethereum is a blockchain platform that implements a Turing-complete programming language, allowing for the existence of code stored in the form of "contracts". This allows for the practical existence of Smart Contracts, a concept first proposed by Nick Szabo in 1996 [18]. According to Szabo, *"a smart contract is a set of promises, specified in digital form, including protocols within which the parties perform on these promises"*.

So, at a more superficial level, a smart contract is just source code, a structure that follows pre-specified rules, in order to move around assets and their ownership, such as cryptocurrency tokens. A smart contract reacts according to the interactions it has with the other elements of the blockchain, which, in a crude manner, can be either people or other contracts. In other words, Ethereum moves beyond the realm of currency and opens up the possibility for decentralized applications to be ran directly on the blockchain. Smart contracts are written in different kinds of custom languages, like Solidity, which are then compiled to byte code, deployed to the chain and interpreted by the Ethereum Virtual Machine (EVM).

Unlike with Bitcoin, Ethereum's blockchain saves, not only the transactions, but also the state of its smart contracts. The smart contract code is also subject to the consensus mechanism, as it is being run on the same nodes that validate the transactions.

Currency

The interesting aspect of Ethereum, however, is that the code from these applications or contracts is executed by the peer-to-peer network nodes, making Ethereum a globally available computer. Of course, with computational power comes a price. Each time a function from a contract is called, someone's computer, a mining node, is doing all the computations, and for each line of code, an agreed fee must be paid. As such, Ethereum has its own cryptocurrency, by the name of Ether, and which is essential to fuel the network.

Consensus

At the moment, Ethereum is still using PoW as the consensus protocol, in a similar fashion to Bitcoin. There are projects currently trying to move Ethereum to PoS, such as Casper [19]. PoS is a consensus protocol with a different paradigm, in which the block mining process is roughly based on trust, on the fact that the miner has a certain stake or investment (like cryptocurrency) in the network, and so it is in their best benefit to be an honest node.

The benefit of PoS over PoW is that there is not as much "waste" of computational power as in PoW. In PoW, miners have to intensively search for a target number, which allows them to claim the block as their own. This serves no other scientific or practical purpose other than making the mining process hard, and, as soon as a node successfully mines a block, all the energy used by all the other mining nodes that were in this process basically goes to waste [20]. There are other concerns that make a PoS system like Casper a better option as well, such as performance concerns.

Performance and Scalability Issues

Ever since its conception, there has been concern as to whether Ethereum's **throughput** and **latency** are enough to handle a large amount of applications running at once, and if it will be scalable in the future. The recent launch of an application by the name of CryptoKitties [1] disrupted Ethereum, congesting the main network and slowing down the speed of transactions, again raising concern over Ethereum's performance.

These concerns are also very much valid for the case of supply chains. If Ethereum were to be integrated with a supply chain, by using smart contracts, one of the most important factors to take into account would be exactly the performance and scalability

[1] https://www.cryptokitties.co/

of the system. It is important to note, though, that private and public blockchains have different performances as well as different scalability concerns. Furthermore, certain values that affect the performance (such as the block time) also have an affect on the scalability, and consequently, on the security (a blockchain with a lot of nodes is more secure than one with fewer, for instance). This means that these attributes are interdependent and require a fine balance. It is important to mention that it is possible to deploy a network for Ethereum separate from the main network, so that it runs on a separate number of nodes and so that the network could have some of its values customized. However, such a network will still be public, like the main network already is, suffering from most of the same issues.

In a public chain, security is a much bigger concern, as there are many different kinds of attacks, and the blockchain's values, such as the block size, are balanced in a way that tries to prevent these. Such values can easily be adjusted in a private chain, in ways they can not in a public chain, where they would cause forks, clog the chain or raise security issues. In addition, private blockchain deployments are usually more centralized, concentrating their power in fewer nodes, as there is an inherent sense of trust from within the network. For this very reason, this type of blockchain is more easily scalable as well.

The performance of blockchains such as Ethereum is usually measured by the average throughput, which is the number of processed transactions per second (TPS). This depends a lot on two main factors:

- The block size - in Ethereum, the block size is not a fixed value; rather, it has a "gas" limit; each transaction put into the block spends some gas, and when the gas reaches the limit for that block, the block is full; here, "gas" is a unit that refers to the computational steps taken to process a certain transaction or contract; the more complex the transaction, the more "gas" it spends;

- The average time to publish a block - block interval; in Bitcoin, this was a fixed 10 minute time, but in Ethereum, this time can be as low as 15 seconds [21]; this is directly related to the latency, the time a transaction takes to be integrated into the blockchain;

The fact that both transactions and block can vary in their size makes it hard to theoretically pinpoint what the performance of an Ethereum network is. In practice, though, and according to recent studies and statistics gathered, the throughput of the main Ethereum network is around 15 TPS.

Besides latency and throughput, most blockchains are facing the problem that the **blockchain size**, the required to store the blockchain itself is becoming increasingly large, and is constantly growing. With this kind of pace, not every node will be able to store the chain in order to validate it.

This easily becomes a security problem for public blockchains, as the power will be concentrated among the few nodes who can store the blockchain. In Ethereum, every node has to store and process all transactions as well as store the entire state of the blockchain [21], hence, this is an important issue to address.

Proposed Solutions

Solutions to the above mentioned issues of the above mentioned issues have already been proposed and are being tested. Some of these are Raiden, Casper, Sharding and Plasma.

Casper - PoS-based efficiency and security scalability To put it simply, Casper is a partial consensus mechanism that aims to implement PoS as the consensus algorithm in Ethereum. It works by allowing anyone who holds anything of value in the system, in this case, the Ethereum currency, Ethers, to participate in choosing the next block.

Casper has the important characteristic of accountability. The validator nodes deposit some of their currency to have a stake on the mining process of a block, so they have their own deposits at stake. Therefore, they are encouraged to be honest, because any violation of the rules or attempts of attack puts them at risk of losing this deposit. The higher the stake, the higher the power: with a high stake, it might be easier to perform an attack, but as it is also a lot riskier, so attacks are discouraged.

Another important aspect of Casper is that, as a PoS based mechanism, it allows for a more efficient Blockchain as it enables miners to work without high end hardware and high electricity costs, while also potentially allowing for lower block times.

Sharding - Transaction throughput scalability Sharding is a solution that aims to increase the throughput of Ethereum. It acts upon the problem that every node must validate every transaction, by dividing nodes and transactions into subsets and assigning them such that a subset of the nodes will verify a subset of transactions [22]. This allows for transactions to be processed in parallel, and, as long as there are enough nodes validating a transaction, security is maintained, and more transactions per second are able to be processed, therefore increasing throughput in a scalable way.

Raiden - Token transfer throughput scalability Raiden is a solution that allows for off-chain transfers of tokens to happen outside the blockchain, on a side-channel, in a secure way, using cryptography to build *balance proofs* [23]. This way, for a certain number of transactions, a side channel can be opened, the transactions are performed, and, at the end, the channel is closed and the transactions are submitted onto the chain. Raiden implements the protocol that provides the routing between the nodes and all other functionalities that make this possible.

Plasma - Throughput scalability Plasma uses smart contracts to build hierarchical side chains, that can be thought of as children of the main blockchain. These chains process information as a normal chain would, being governed by their own rules and constraints, and it is even possible to revert transactions within them, as they have a smaller scope and are built apart from the main chain. These child chains can then relay information back to the main blockchain. Theoretically, there is no limit to the number of chains, allowing for great scalability [24].

2.4.2 Hyperledger Fabric

Hyperledger is an open-source effort hosted by The Linux Foundation, which aims to further improve the concept of blockchain to be used in many different contexts.

More specifically, one of the Hyperledger projects, Hyperledger Fabric, initially contributed by IBM, tries to address some of the open issues of the existing frameworks, like Ethereum. Hyperledger Fabric is an open source framework to build permissioned (private or consortium) blockchains usable in industry, and, while it shares some similarities with Ethereum, it also has many differences.

Chaincode

Hyperledger possesses a form of smart contracts, which it names *chaincode*. Similarly to Ethereum, the chaincode programs are event driven programs which run on the ledger, changing the ledger's state when executed, thus being able to move assets around. It can be said that this works much like a state machine.

A big difference to Ethereum, though, is the language used in the chaincode. Programs are written in Go, with support for other languages like Java to be implemented. The code is deployed using an isolated VM, through a Docker container.

Being able to use general-purpose languages to code smart contracts is an advantage and lowers the barrier of entry to use this framework as a blockchain. But it also raises the issue that the code might be non-deterministic, leading to forks, which is the specific reason why Ethereum uses a language that it compiles and runs on EVM.

Architectural Revision

The latest version of Fabric, v1.1, has a totally revised architecture from some of the previous versions, like v0.6. These changes in architecture originated from changes in the requirements, as well as to better address some other underlying issues of the technology.

The previous version (v0.6) shared similarities with other blockchain systems, in that it had an order-execute architecture, which combines consensus and execution of transactions, in this order. In the order-execute architecture, after a block is assembled and mined (agreed upon), all other nodes will execute its transactions sequentially. This was a simple architecture, but it has several liabilities when used in a permissioned blockchain like Hyperledger. Some of the most notable issues are:

- Non-deterministic code: In this architecture, any non-deterministic code might cause forks, because the code might change the state in different ways for different nodes;
- Sequential execution of smart-contracts: one slow contract could cause a delay for the others;
- Hard-coded consensus: there is no "one-size-fits-all" solution, protocols vary immensely in performance and functionality, and having alternatives and adaptation to specific use cases is essential;
- Confidentiality of execution: sometimes, smart contract logic is intended to be restricted, only to be run by some nodes, though the state should be propagated to all nodes in the end; this architecture does not allow for such a restriction

A detailed architectural specification of Fabric's revised version 1.1 can be found in the documentation section of Fabric's website [25] [26]. Other articles have been produced by IBM researchers [27] [28], as well as numerous presentations, detailing the general architectural problems and solutions of this technology.

This revised version follows instead an "Execute-Order-Validate" implementation, which tackles the above mentioned issues, and, according to Fabric's documentation, the advantages this reviewed architecture brings are **scalability**, **confidentiality**, **consensus modularity** and **chaincode trust flexibility**.

System Architecture and Consensus

In Fabric, transactions are of two types: either **deployment transactions**, which create new chaincode, or **invoke transactions**, which perform operations on the deployed chaincode.

Being a permissioned blockchain, any node which wants to interact with transactions is required to authenticate itself and have an identity. Transactions and data are therefore restricted to certain participants of the network. The data partitioning mechanism used to control this is called a **channel**. Users can belong to a channel, having visibility of these transactions, and the consensus will happen only within the channel and its members. This also means that there is one ledger (set of transactions plus the state) per channel. The channels themselves have defined sets of rules for what actions each participant can execute.

Besides the existence of permissioned channels, the other main difference in this revised architecture is the way in which consensus works. The following definitions correspond to steps in the consensus mechanism, which are needed to better understand this consensus mechanic.

Endorsement Some endorser stakeholder first verifies the transaction and then decides whether to accept it or reject it. The endorsement is simply a signature of the transaction, which confirms the decision of the endorser. Endorsement policies might require transactions to have a certain minimum number of endorsers to be accepted.

Ordering All the transactions gathered within a certain period are grouped into a block, sorted and committed in that order.

Validation Check if the endorsement policy of a transaction is satisfied and then checking if the transaction transformation is valid.

The consensus process is held by three different types of nodes:

Types of nodes:

- Client - submits a transaction to the endorser nodes
- Peer Node - peers form a peer-to-peer gossip network, maintain the state of the ledger and manage the chaincode; they can be
  - Endorser - simulates transaction execution and decides whether to endorse or not. Endorsers are always also committers.
  - Committer - verifies the endorsements and validates the transaction results
- Orderer Node - the orderer nodes together form a ordering service; this service uses a certain pluggable (changeable) protocol to order the transactions received from the peers, create a block with them and then broadcast this block to the committers;

In a simplified way, the workflow of the consensus, as shown in Figure 2.2, is as follows:

1. The client transmits the transaction to the endorser nodes.
2. The endorser nodes simulate the transaction and choose whether they endorse it or not. If they do, they sign the transaction and send the endorsement back to the client.

3. The client broadcasts the endorsement to the ordering service.

4. The ordering service gathers incoming transactions, sorts them into blocks and then broadcasts the transactions by order to all the peers (orderers and committers); the peers then validate the transactions and verify their endorsements, applying only the transactions which fulfill the endorsement policy.

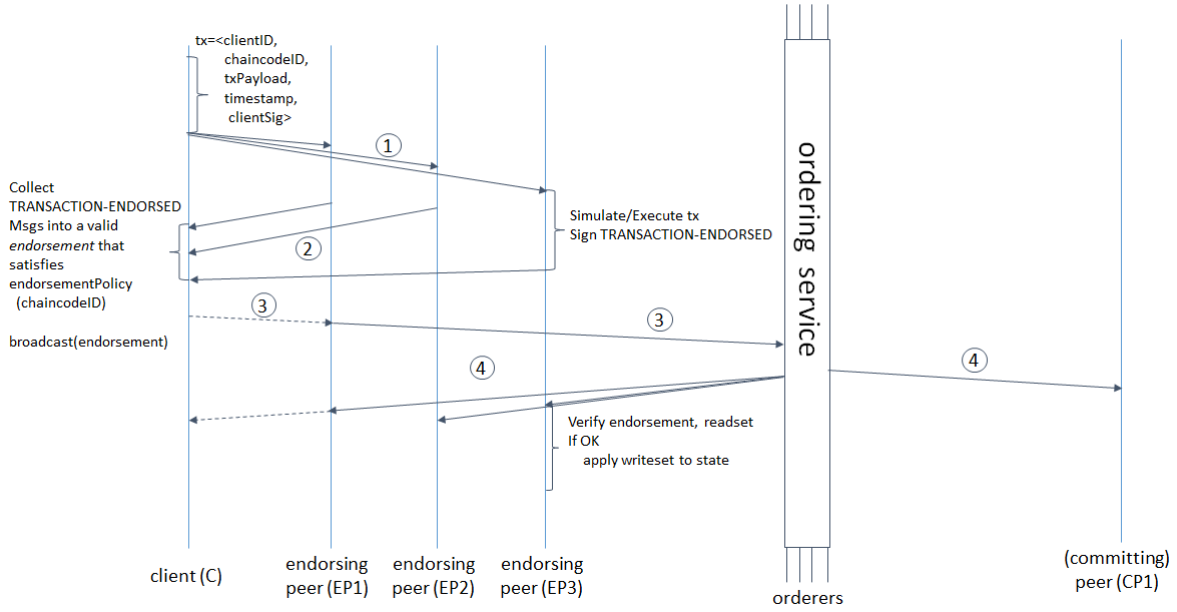


Figure 2.2: Consensus workflow in Hyperledger Fabric. Available in [26].

Compared to other blockchains, this consensus mechanism has some advantages, though its application is somewhat limited to some use cases. Table 2.2 summarizes some points of difference between a blockchain using Proof of work and one, like Hyperledger, using a BFT state machine consensus.

In brief conclusion, Hyperledger allows for a permissioned blockchain which sacrifices some decentralization for scalability and performance. By using a different consensus mechanism, most of the issues of public blockchains vanish. Furthermore, data access must be authorized, such that sensitive data in a channel is protected, one of the requirements for data in supply chains. Thus, Hyperledger Fabric's features are adequate for SCM's requirements in a way that is difficult to achieve in public blockchains.

2.4.3 Hyperledger Composer

Hyperledger Composer is another tool which works together with Fabric to allow for an easier approach to modeling a blockchain network. It is actually a higher level

Table 2.2: Comparison between traditional PoW consensus and Hyperledger's BFT consensus mechanism

	Proof of Work (Bitcoin, Ethereum, ...)	**BFT state machine replication (Hyperledger, Ripple, ...)**
Membership type	Permissionless	Permissioned
User IDs (Sybil attack)	Decentralized, Anonymous (Decentralized protection by PoW compute/hash power)	Centralized, all Nodes know all other Nodes (identity management protects against Sybil attacks)
Scalability (no. of nodes)	Excellent, >100k nodes	Can scale to 100 nodes, with certain performance degradation
Peak Throughput	from 7tx/sec (Bitcoin) and 15tx/sec (Public Ethereum)	>10k tx/sec with existing implementation in software [>10 nodes]
Power efficiency	>1 GW (Bitcoin)	Good (commodity hardware)
Temporary forks in blockchain	Possible (leads to double-spending attacks)	Not possible
Consensus Finality	No	Yes

abstraction of Fabric, which allows for a big reduction in the lines of code written and simplifies the whole process of building a business network and deploying it, as well as the process of managing the identities of the network's participants. The whole documentation for the project is continuously being updated at their website [29].

It features a nice and intuitive graphical user interface to manage all of its components, and it also features a REST server to expose the blockchain to outside applications.

Business Network

A business network is a model of all the data in the blockchain, which includes all the objects, functions, transactions and identities that will connect to each other and be saved on the ledger. It is basically an abstraction of the chaincode that will be installed onto Fabric. A business network, being a network model, can be deployed into an instance that runs on a certain number of nodes.

Composer features a custom modeling language which makes it easier to define a business network. It is divided into 4 main components, as can be seen in Figure 2.3.

If a business network model is deployed multiple times with a different sets of nodes each time, it will simply constitute different instances of a network model. Each instance will constitute a separate network with a separate ledger, independent

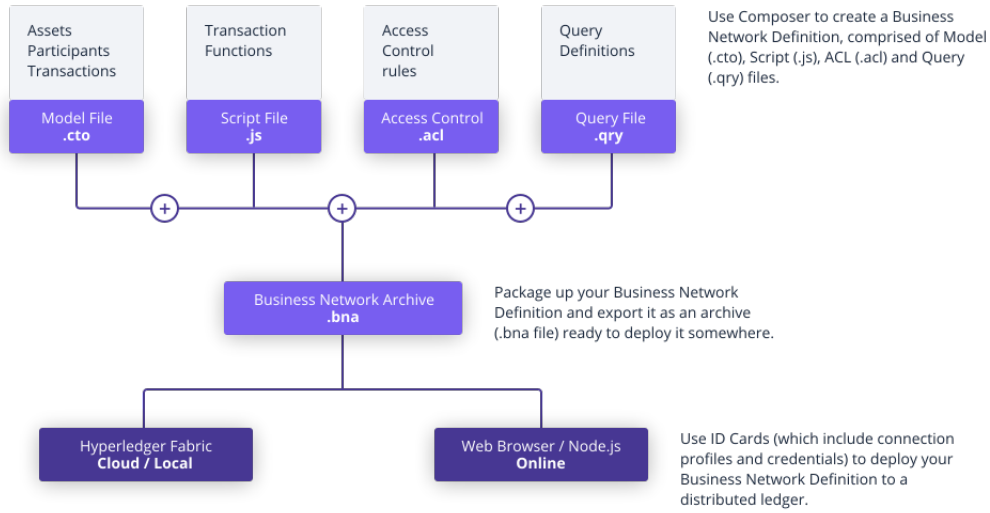


Figure 2.3: Hyperledger Composer abstraction over Fabric. Available in Composer's documents [30].

from the other networks. This is analog to modelling a class in a object oriented programming and then instancing it multiple times. Each instance is separate from the others, even though it is based on the same model. Therefore, a good model design can be used by several businesses separately, and each can configure the model to represent the topology they are looking for.

The 4 main components:

Model File This is where all the main data components are defined. **Assets** are the main "objects" that are handled in transactions, and they can represent a variety of things. One example can be a "car". **Participants** are the types of people that will be participating in the network. One example participant can be "Customer", which will have fields like "Balance" and "Name". An instance of a participant can be linked to a real identity of a person using the blockchain. **Transactions** are the most important data object. They model what actions can be taken, and they usually have some data fields associated, which act as the input of the transaction. An example of a transaction can be "BuyCar", which could have as parameters the car ID, and a participant could then call this transaction, if they had the permission to do so. All these 3 components, Assets, Participants and Transactions have their own registry. While the Assets and Participants registries are mutable, the Transactions registry is not. Additionally, another type of object can be defined in this file, **Events**, which can

be emitted by transactions and subscribed by applications which can handle them, for instance, to generate notifications.

Script File This file contains all the functions that define how a transaction behaves and what kind of data it processes and outputs. It can also have additional auxiliary functions.

Access Control This is where the access rules are defined. These rules include restrictions on who can invoke which transactions, who can read the data, and many other permissions.

Query File This file includes some queries that can retrieve information from the blockchain. This information might include any assets, participants or transaction history, and it is also subject to the access control rules. These queries are both exposed to outside applications through the composer REST server and to the Script file functions.

All of these components can be compiled into one file (.bna), which is then deployed onto Fabric.

Authentication and Identities

Hyperledger Composer also features identity management. An identity is a digital certificate and private key, in the form of an account which can interact with the blockchain network. Within a certain business network, an identity can even be bound to a certain instance of a Participant type. For example, if there is a participant type called "Customer", an instance of that participant called "Customer1" can be created, which can then be bound to some real person's account. That person will then be able to interact with the blockchain, but will be bound by whatever access rules the participant "Customer" is bound by.

In order to make identity management easier, Composer has a feature called "Business Network Cards". These cards are files which can are associated with an identity and a network, and also act as a private key. If a user possesses a card, they can import it to prove their identity and start interacting with the blockchain.

Fabric Topology

Composer also has configuration files for some of Fabric's properties. It possesses files to determine which organizations exist and which kind of nodes they possess, along with their mapping. It also has a definition for the endorsement policies.

2.4.4 Corda

Corda's take on blockchains vastly differs from Ethereum or Hyperledger. Traditionally, banks have their own ledger, but Corda wants to unify these ledger's into one, claiming that it improves the cost and efficiency. Corda focuses solely on providing a distributed ledger for financial services, where financial agreements can be recorded and managed, available to any financial institution[31].

This is a very specific purpose, which means that Corda's application on Supply Chains would be rather limited, to fixing payments and agreements, perhaps. Even Corda's own smart contracts are pretty limited in what they allow, in comparison to Ethereum's, since they do not allow for all the freedom of computation. In theory, Corda is limited to the financial and legal contracts domain, with the functionality for custom contracts being very limited.

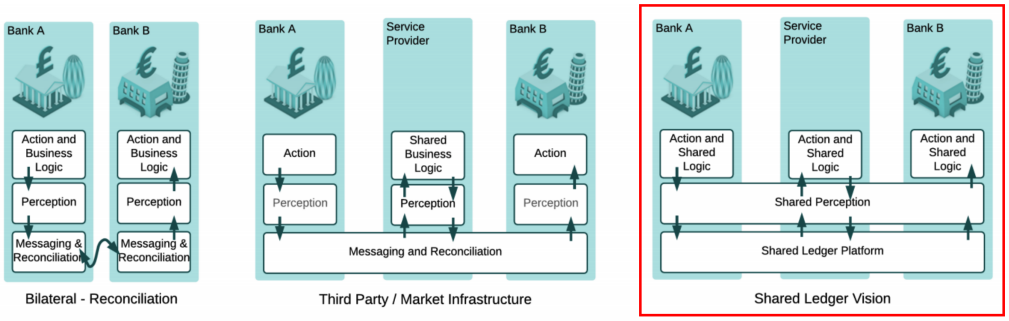


Figure 2.4: Corda's vision on a Shared Ledger. Available in Corda's White Paper [31].

Concepts

Corda is also a **permissioned blockchain**. While the ledger is global, some transactions and agreements involve only certain groups and should be visible only to the participating parties or those with a reason to be able to see them.

There are some other important concepts to understand how Corda works:

State Objects The state of the ledger, where the agreements are set and stored, in the code from the contracts.

Transactions These are the actions that change the state of the ledger.

Transaction Protocols It is the business logic that enables the coordination of actions, through the smart contract code.

CorDapps

CorDapps are distributed applications running on the Corda platform. The concepts of State Objects, Transaction Protocols, among others, all fit into these distributed applications.

The functionality of these applications is, however, rather limited, since a CorDapp's ultimate goal is to program a specific agreement, which will then lead to updates on the ledger. So, similarly to a "smart contract", the execution of CorDapp's leads to transactions which change the state of the ledger, fixing immutable agreements into the blockchain.

These applications have code, just like a smart contract would. But, while a normal smart contract is just code, a program, CorDapps have another side to them: they incorporate legal prose into the code, transforming them into something akin of a "smart legal contract", legally enforceable contracts, clearly designed for the highly regulated environments of the financial industry.

As with Ethereum, Corda also uses a virtual machine. In this case it is the Java Virtual Machine (JVM), and the applications are written in Java.

2.4.5 Comparison

Comparison's between these three platforms are recurrent, though Hyperledger and Corda might have more in common with each other than with Ethereum. Valenta [32] concisely describes the main points of difference between these frameworks, with a focus on the consensus process and general characteristics of the frameworks.

He concluded that Ethereum is more of a general purpose platform, while Fabric tries a different solution, adaptable to a different set of problems. Fabric is flexible and customizable in ways that the other frameworks are not. On the other end, Corda was designed with one specific application in mind, financial services, and it is rigid in that sense, which actually made its architecture a lot more simple compared to

Fabric's. It is also arguable that Hyperledger can be tailored to resemble Corda, due to its modularity, as Corda is not really a competitor of Hyperledger, but rather a complement.

Hyperledger vs Corda In one sense, Corda has more of an out-of-the box experience, while with Fabric, the logic would have to be implemented. Composer does make Fabric more user-friendly, but it is still hard to setup and configure, compared to Corda. It all comes down to what kind of system we want to build, and what are its the final objectives, functionalities and requirements.

Ethereum vs Hyperledger Also, as already pointed out in Section 2.4.2, a permissioned ledger like in Hyperledger might better fit the supply chain industry requirements than Ethereum, though one solution does not discard the other. Another important aspect to distinguish is also the difference in performance between these platforms.

Performance

The difference in performance of various blockchain platforms is a crucial topic to address, that might lead to choosing one platform over another, with similar functionalities. In the case of Supply Chain, it might be important to measure the difference in performance of Ethereum and Hyperledger, for instance.

Pongnumkul has conducted a performance analysis of private deployments of these platforms [33] and he reached some interesting conclusions. In summary, **Hyperledger's performance is higher in every aspect than Ethereum's (on a private deployment)**, but, for the same resources, **Ethereum can actually handle a higher number of concurrent transactions**.

The metrics used in the analysis are **throughput**, **latency** and **execution** time. In every test performed, Hyperledger had a higher throughput, lower latency and lower execution time than Ethereum. The disparity between these numbers also grew with the number of transactions, with Ethereum's performance degrading much more than Hyperledger's for the same increase in transactions. This means that Hyperledger not only performs better, it also scales better, as seen in Figure 2.5.

Another interesting conclusion was that, with a really high number of transactions (10000), Hyperledger only took 3.57 seconds to finish executing the first transaction, while Ethereum took 361.36 seconds, showing that Ethereum's network clogs easily

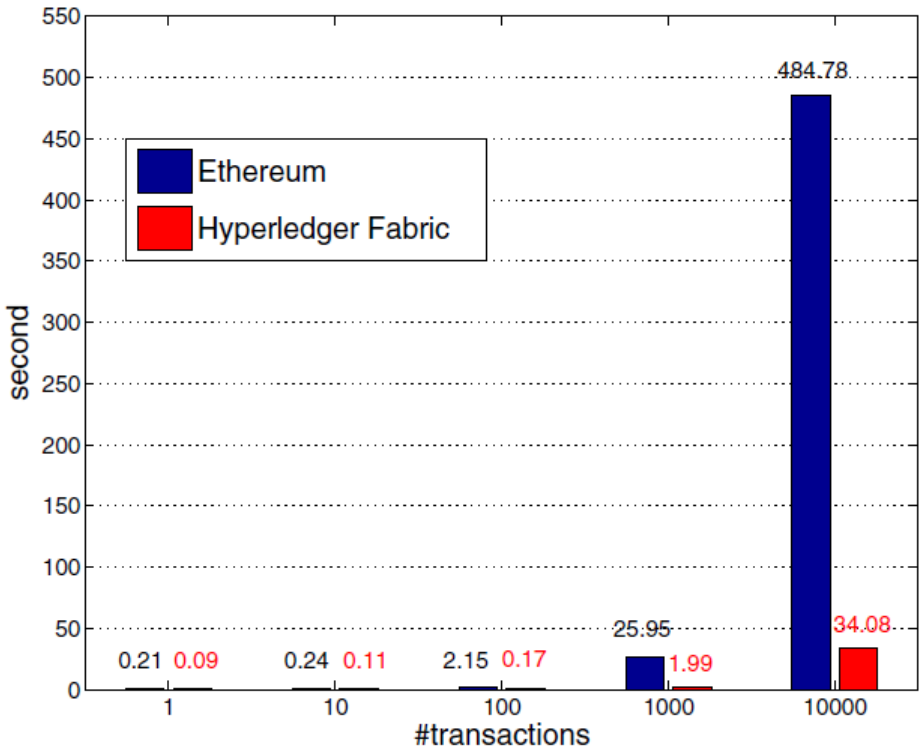


Figure 2.5: Comparison of average latency between Ethereum and Hyperledger, with growing number of transactions. Available in [33].

and is slow to start up, while Hyperledger executes the transactions more consistently. This was exactly one of the reasons for the revision in the architecture of Fabric.

Chapter 3

Blockchain in the Industry

This chapter presents an overview of how the blockchain might apply to the industry in general and supply chain in particular. It goes on to describe the general advantages of using blockchain, with a focus on how it might positively affect SCM, as well as the disadvantages and challenges the integration of blockchain might face. Some applications that are already trying to merge the concepts of blockchain and Supply Chain are presented. Finally, an overview of some design alternatives are given, which will be the basis for the design analysis and decisions later on.

The purpose of this chapter is to present some topics that might unify the topics of blockchain and supply chain into one, to ascertain if they are a good fit.

3.1 Advantages of Blockchain in Supply Chain

Some of the advantages blockchain would bring to supply chain, over other solutions:

- **Less error prone**: reduction in errors on manual data entries, especially when combined with IoT and other automated processes;
- **Enhanced security of transactions**: not only is the ledger immutable, attempts at fraud are easily detected.
- **Improved tracking**: the ledger is easy to analyze and delivers the results really fast, making it possible to know the status of any order or asset at any time; at the same time, any error, either accidental or on purpose, that manages to find its way into the system is easily traceable.

- **Improved consumer trust**: blockchain could allow users to check the provenance of their products, developing a relationship of trust with the suppliers.
- **Reduced costs**: **reduced governance costs** for exchanging info and etc, allowing for higher efficiency and faster times at processing the information (enhancing cost effectiveness); **reduced internal management costs**, increasing efficiency and sustaining competitiveness; **reduced product or service costs**, creating competitive advantage and barriers to competition, reduced supply chain lead times and increased flexibility in supply chain design.
- **Internal supply chain trust**: It is important that the elements of a supply chain trust the information that comes and goes from each other, and blockchain allows this to happen.

This last point is one of the most important aspects, which is often overlooked in favor of other more obvious functionalities.

Informally speaking, a business knows it can easily access its own data at any time, so there is usually a lot of self trust. Therefore, businesses trust and rely a lot on themselves, but not always on others. Trust means being cooperative and relying on other business for the needed data, and believing that this data is also error-free.

As described by Panayides [34], co-operation and trust are the key in improving supply chain performance and innovativeness, increasing the quality and leading to benefits for all parties involved. Similarly, Yeung [35] managed to find a relation between trust and a higher supply chain integration. In the context of supply chains, **trust might be defined as not only loyalty, but also reliability**. This last aspect is very important, because it measures just how much you can expect from your partners in a supply chain. And when you need information quickly, trust in the form of reliability is a very important asset to have.

In this sense, if the blockchain technology manages to improve the information flow in a supply chain, while maintaining security and trust between parties (at least at a technical level), then it follows that, just as concluded by Panayides and Yeung, the supply chain itself will have an improvement in performance, since the parties involved do not have to worry as much about these aspects or any power struggles. Therefore, **trust seems to be a key factor in building an efficient and effective supply chain**.

3.2 Challenges of Blockchain Application to Supply Chain

The reverse side of the coin is that Blockchain is not always as good a solution as it is prophesied. Blockchain itself is a topic in research and, while some of its applications and advantages are quite obvious, many of its disadvantages are overlooked, and these might be crucial when making the decision to apply blockchain to a supply chain.

3.2.1 Technical Limitations and Scalability Concerns

The technical limitations include, but are not limited to: throughput, latency, size and bandwidth and security.

- **Throughput**: Current blockchain technologies, even in private deployments, such as Hyperledger, have a high throughput, but not as high as certain centralized systems. This is one of the main concerns in supply chain, that a blockchain can not process the information quite as fast as the current systems, which could lead to a decrease in performance and further delays. As in any distributed system, though, the decrease in speed is often the price to pay for decentralization in supply chain. Ultimately, while the flow of information inside one specific company might be lower than before, the flow between different companies, which were previously not integrated, might actually be much faster than before. It is not fair to compare the speed of a centralized system with the speed of a distributed ledger, since the latter offers more functionality and disperses the information further where the former could not. A slower dispersion of information globally, through various entities, is, in this case, preferred to a fast dispersion only locally, through an organization's system or its closest associates. In conclusion, there might be a tradeoff between throughput and functionality, with higher functionality and integration being achieved at the cost of throughput.

- **Latency**: Similarly to throughput, and related to it as well, the latency of a transaction is something to take into account. With some blockchain deployments, like Bitcoin, each transaction might take a long time to validate, and this time might depend on the fees paid as well. This is mitigated by permissioned platforms such as Hyperledger, which have low latency, even in the presence of a high number of transactions, and possess no currency or fees to worry about.

- **Size**: The more transactions are processed and information stored in the blockchain, the bigger it actually grows. In the current context, if we were to deploy a global blockchain for all the supply chains, it would probably grow way too large in a small amount of time, which would not be sustainable in the long run. In a more limited scope, however, it would probably not be as big of a problem. There is also a lot of research in the optimization of blockchain size.

- **Security**: One concern for blockchains in general is how security is handled. This issue is more important in the case of public blockchains with PoW consensus. In the case of supply chains, there are many alternatives, and possibly having a public PoW blockchain is not the optimal one, so this is not the main security aspect to worry about. There is, however, another aspect to take into account, which is the fact that the hash functions being used at the moment might be broken in some years. If this were to happen, the immutability property of a blockchain would be broken and any prospects of proving the provenance of products or their traceability would lose their groundwork.

3.2.2 Lack of Interoperability Standards

Provided that there is a way to share information, companies each have their own means of inserting information into their systems, in whichever format they want, as long as the correct pieces of information are there. Other companies which might want to access this information might have a difficult time understanding just what to look for and where to look for it.

This is the first part of the problem, the lack of standards by which companies should abide to, if they want to form some common ground by which they can cooperate and understand each other's information.

The second part of the problem deals, in a more technical way, with the lack of interoperability in the system's themselves. Many Enterprise Resource Planning (ERP) systems are operated under closed environments, with information often being manually entered into the system, and no APIs for external systems to connect to.

3.3 Similar existing applications

Some projects which try to fit blockchain as a solution for improving SCM have already emerged, or are in being worked on. This section explores some of these applications.

3.3.1 CargoX

CargoX delivers a solution for making the Bill of Lading (B/L) documents digital. To give some context: in a supply chain, many times, the products are delivered by cargo ships, inside containers. The B/L document has the same value as the value of the goods that are declared on it, serving the following functions:

- It is a receipt that acknowledges the loading of the products.
- It contains the terms of the contract of carriage.
- It is a title to the goods it declares.

These characteristics make it an extremely valuable document, which must be transferred from the carrier to the company acquiring the products in a safe way. Losing this document would mean losing the rights to all the goods in the shipment, as well as losing the proof that they were even shipped in the first place.

CargoX uses Ethereum's smart contracts to put these paper documents into the blockchain. It has a built in token system that allows to exchange the document's ownership immediately after payment [36].

Relevance and Applicability

It is a very specific solution for a very specific problem. It is a concept that could be integrated in a broader solution, and should fit the needs of this dissertation partially, in theory.

The execution of the concept, however, seems to be lackluster in that it uses a public blockchain, Ethereum, for extremely valuable documents. One one hand, might bring about security issues, or even other problems. If Ethereum were to suffer a 51% attack or any other kind of attack, the whole project would be compromised, along with the bills of lading.

On the other hand, the project uses a token system that seems designed to monetize a concept and also to move currency around more than the documents themselves.

In the end, the idea of transmitting documents is an integral part of information integration, especially when these documents prove the ownership of something, and blockchain, depending on the execution, might be a good idea to digitize that proof of ownership in a way that can be traced.

3.3.2 Eximchain

Eximchain is an all-round solution, which acts as a ledger, recording historical data and transactions, as an inventory management tool and also provides financial applications, by means of smart contracts. It was developed using a fork of Ethereum, Quorum, which is a permissioned version of Ethereum, focused on enterprise use, which means Eximchain runs on its own network [37].

The smart contracts of Eximchain allow for the transmission of finances and verification of the validity of orders placed. All the data from these transactions, which include the transmission of goods, is recorded onto the ledger, allowing suppliers to prove their reliability to deliver.

At the same time, the inventory, as well as all the goods, as are tracked across the supply chain with small delay, with the information being accessible seamlessly among partners. This allows for more accurate supply and demand predictions, especially if the information is directly integrated with forecasting systems.

Relevance and Applicability

This application seems to focus on the most important points of supply chain management: the tracking of goods by the enterprises, the financial process of transmitting those goods and the integration of this information. These applications are highly practical and resonates with the goals that this dissertation tries to achieve, which are exactly to turn SCM more efficient by making the information readily available.

3.3.3 OriginTrail

Like CargoX, OriginTrail operates on the Ethereum network, and has its own ERC20 tokens. It synchronizes all supply chain data in the platform, and uses these tokens as an incentive. Underlying the use of the Ethereum network, OriginTrail has a custom network topology protocol designed as a privacy-layer, which can be seen in Figure 3.1. This network uses zero-knowledge methods to validate data without making it accessible. This custom protocol consists of Data Holder (DH) and Data Creator (DC) nodes working together to receive, transmit, store and process the information. The data is communicated to the blockchain, where people with OriginTrail tokens can interact with it.

The main objective of the project is to ensure traceability of the products as they move from supplier to retailer, all while ensuring the data is not tampered with, which

means this project has a high focus on privacy. The tokens serve as a way to exchange data ownership, as well as to make reviews on a reputation system [38].

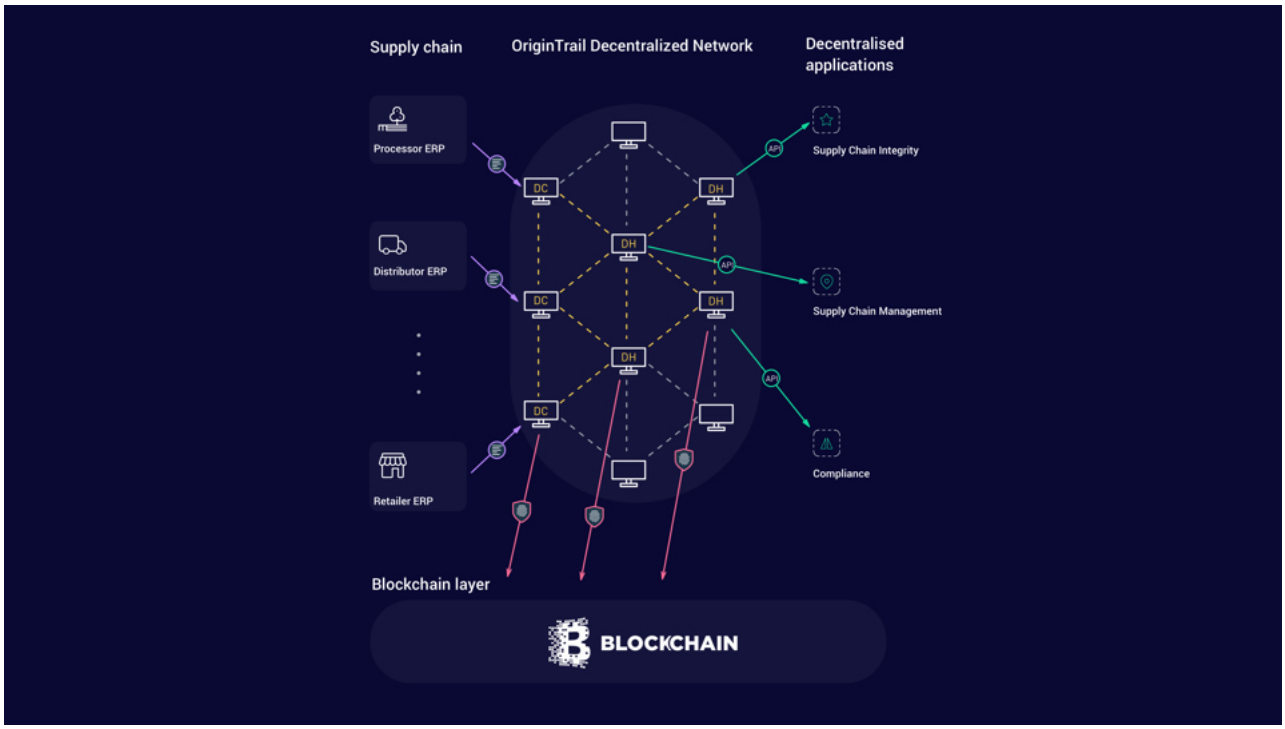


Figure 3.1: Overview of the OriginTrail network. Available in [39].

Relevance and Applicability

This solution specializes more in the tracking of the goods, especially so that the public can know where their products are coming from, as well as the companies. This allows for higher integrity and authenticity of the products. A good example of a use case is in situations where a bad product is detected and batch recalls need to be made. With this tracking, a specific batch can be identified and removed, whereas without the tracking, product providers as well as the product buyers would have to get rid of substantially higher amounts of that product, leading to a loss of revenue.

Though it is a more specialized solution, it is important not to ignore the aspect it focuses on. This can be one of the easiest methods to save money by integrating the supply chain with the blockchain.

3.3.4 Ambrosus

Ambrosus focuses on tracking the quality of the goods transported during shipments, specializing in the food and pharmaceutical industries. Not only does it track ship-

ments throughout the entire supply chain lifecycle, it also uses the Internet-of-Things (IoT), through sensors, to communicate quality metrics of the products in real-time to the blockchain.

The enterprises can then query this data and retrieve it immediately to their own databases, effectively achieving a better quality assurance. It is also provides enforceable proof of the quality or, if otherwise, the lack of it.

Like OriginTrail, Ambrosus also has its own protocol, called AMB-NET, which hosts smart-contracts and communicates with the Ethereum blockchain.

Relevance and Applicability

This is another project with a more narrow scope, specializing in the tracking of the quality by using IoT. It is something that the other projects are not implementing, but it is important nonetheless. To some industries, it is even more important than any other feature, which makes quality-tracking a **must** in a supply chain management blockchain project.

3.3.5 IBM and Maersk's Demo

IBM, together with Maersk, just recently launched an innovative project which aims to create an open platform, using Hyperledger, for information sharing on a large scale [40].

It is an open platform, which features a shipping information pipeline and paperless trade. It deals mostly with the transportation of goods and automating all the processes associated with it.

Relevance and Applicability

Although it has a similar purpose to the one of this thesis, there have been reports of companies discontent with this platform, since it does not focus on the industry as a whole and ensure common standards. Instead, it is a platform designed to meet IBM and Maersk's expectations. Other than this, there has been very little information about it yet or about any progress made [41].

3.4 Designing a Blockchain-based Supply Chain

Blockchain is not always a one-size-fits-all solution, and its use must be carefully tailored to the application in question and to the specific requirements. This section describes some important points of focus to have in mind when making decisions for the design of a blockchain based supply chain.

3.4.1 Integration Models

Point-to-point Business-to-Business (B2B) Data Interchange - The integration must be designed between each two specific endpoints. Each new connection must be modeled separately. In a large scale, this does not work very well, it is a model that only works under specific cases and requirements, since it requires a customized integration.

One-to-many entities Hub B2B - A company can develop a connection endpoint to which other companies can connect, as long as they follow the hub's communication standards or use its API. This way, a single company can establish connections with multiple intermediaries.

Many-to-many entities Cloud B2B - This model encompasses full integration, where the information can flow freely between businesses. This is the ultimate goal of a public blockchain, but it would require companies the development of interoperability standards, which are, at the moment, lacking. Otherwise, it is the most cost-effect model and the one which can bring about the most benefits as well, given that the companies can develop their services to be integrated with the blockchain.

3.4.2 Key Implementation Components and Features

As seen in the previously mentioned projects, like CargoX, Eximchain and OriginTrail, each of them followed their own purpose and goals, and each of them has a different set of features that allows them to achieve these goals. Similarly, these are some of the key components that a blockchain possesses, and the respective features that must be decided upon:

Information Storage The most basic and important feature of a blockchain is the ability to save data, which is then considered immutable, as well as registering any important events. As such, information storage might be important in the supply chain.

It also makes Inventory Management possible (though its implementation is out of the blockchain's scope), traceability and provenance of products.

Ledger and Transactions Similarly, allowing for transactions and their recording onto the chain might be important, especially in what accounts for payments between businesses.

Smart Contracts Finally, smart contracts have the potential to be an important part of SCM. In the applications we described, smart contracts were used to transfer ownership of either data or products through tokens. This is just one of the many possible uses. Other smart contract uses include tracking items by their location or condition, automatically updating the status on the blockchain and reacting to any important events by notifying the organization responsible for the items. Another possible functionality is the automation of payments upon delivery. In the end, smart contracts allow for virtually any application, since they are code, programs being run on the blockchain, and so, they are one of the components of blockchain with the most potential for new and innovative features to be developed upon.

Chapter 4

Problem Statement

This chapter focuses on explaining the objectives of this thesis, by defining the thesis statement, as well as the approach that will be taken to determine whether the statements and underlying assumptions are valid or not.

After having previously mentioned some background information about Blockchain technology and frameworks, as well as about supply chain management and supply chain issues, it is now time to focus on what these issues mean. Therefore, a possible way to overcome these issues through the means of Blockchain technology shall be formulated and tested.

4.1 Objectives

This section introduces the focus points of the research content and the objectives of the research work.

4.1.1 Main Points of Focus

In SCM, a product's life cycle can be roughly divided into the many phases the product goes through, down from the raw materials up until the finished product ends in the hands of a consumer. Starting in the raw materials, most products undergo iterations of processing and shipment, traveling from one place to another, while being transformed in successively more refined versions and changing owners. This holds true for any product in any kind of industry, and even the simplest of products which might not require any processing (for instance, fresh produce), have to be shipped from their place of origin to the place where they will be sold.

The improvement of the management of this life cycle is one of the main objectives of this thesis. This improvement, however, has many points of focus. Some were already briefly introduced in Sections 1.1 and 1.2, from which the following points are summarized as being important:

- **Speed of delivery** The effects of the evolution of SCM throughout the last century are visible to everyone. Products are bought and shipped from one side of the globe to the other in a matter of weeks or sometimes even days. Whether this is fast enough or not is a question that can not be entirely answered in one collective voice.

  The world around us moves quickly, and in the freneticism and frenzy of our lives, it might happen that sometimes weeks or days are not enough. **The faster the products arrive to their buyer, the faster the buyer can satisfy their needs.** This holds true not only for the final customer of a product, but also for any enterprise that provides products and services to other enterprises, be it in the role of supplier, manufacturers, distributors or retailers.

- **Synchronization** Many times, the data from a company is synchronized with its own servers and software, in protocols and data formats that can only be understood by that specific piece of software. If many companies share this same software, then they can easily integrate the information between themselves.

  The real problem occurs when the companies have no common ground and the data is not transmittable in an automatic way, leading to a lot of unnecessary manual work to export the data from one system and import it into another. Though there may be many causes for this, the logic assumption would be that this problem may be originated mainly from the following:

  - **Lack of development of data integration standards in the supply chain industry.**
  - **Lack of a common technology to store all data, from where each company could have their own software extract the information from.**

- **Tracking** During a product's lifetime, a lot of alterations occur and, sometimes, the records about the origins of the products are lost, falsified, or flat out not kept in a registry. This leads to unreliability in the goods the consumers use everyday and it may happen that some products are falsified and not the real product they were advertised to be. Additionally, it can also happen that the products, not

being properly tracked, do not hold up to the conditions or quality standards that are required by the regulatory entities. This can sometimes even lead to safety hazards. If a product is subject to hazardous conditions during any part of its lifecycle, it may become dangerous to be used or consumed.

For this reason, it is of the utmost importance that the products are tracked since their origin, right up until the delivery to the final customer, as well as the conditions they are shipped in and the transformations they suffer.

- **Security** This point is one of the most important to deal with, as security is comprised of many aspects, such as: who to authorize to access the information and how to restrict this, what authentication methods should be used, how to accurately detect and prevent fraud, etc.

  Information in a supply chain is highly sensitive and it should be controlled so that only trusted entities can access it. Most enterprises (or groups of enterprises) compete amongst themselves to make the most sales, the most deliveries and have the fastest product cycles. **Therefore, the information that is generated in the process of managing a supply chain might be too sensitive to share, in order to keep the edge on the competition**. Additionally, **the information that is generated and inserted into any system should always be verified, in order to prevent both human error and fraud attempts**.

4.1.2 Possible Solutions

While these previous points of focus are not problems in themselves, their improvement will greatly benefit any industry as well as the consumers of these industries. As such, the actual problem here can be narrowed down to **finding a way to satisfy these needs for improvement through technological means.**

Some of the most common and traditional solutions for this problem include distributed systems, and there are many solutions already available. However, these solutions, which include distributed systems like the cloud or distributed databases, might not satisfy all the needs of the supply chain. Researching alternative designs can be a good way to find new and useful implementations that might possibly even revolutionize this area.

So, is there any other tool or technology that can be applied to SCM, in order to satisfy the need for improvement, one that is scalable, as well as secure? In many ways, Blockchain is similar to a distributed database, and it can even be integrated with the

cloud for a mixed solution (the best of both worlds?), so there is a need to investigate if it might or might not be a good solution for the supply chain requirements.

4.1.3 Thesis Statement

The main objective has already been defined: to improve the security traits, tracking of goods and processes, information synchronization, and possibly speed of delivery of the supply chain, along with other minor attributes. A seemingly good alternative that stands out is Blockchain.

Therefore, the main statement that this thesis defends is:

"Blockchain is a good architectural design for the Supply Chain Management domain."

What remains to be answered, though, is if mentioned attributes are really the right requirements to focus on, and if Blockchain can really fill them, as a distributed architecture. Thus, it can be seen that this statement might be based on some assumptions, which gives rise to some more questions. For instance, we are assuming that the points of focus from Section 4.1.1 are really the most important ones, given the background information that was collected.

Thus, the questions that need to be answered first, in order to reach a conclusion towards the main thesis statement are the following:

1. **What supply chain issues, improvements and requirements do the experts really find the most important?**
2. **What is the Blockchain tool or framework most adequate to the development of an architecture that can support these requirements?**
3. **Is it possible to build a feasible architectural design, by using such a tool, to implement all these requirements?**

These questions are sequential, meaning that, in theory, in order to answer one of them, we have to answer the previous question. But in practice, it is not always practical to follow this sequence, and a question might serve as a guide to start working on the following one. Therefore, the questions can be worked on iteratively. For instance, having some answers to the survey might allow work on the architecture and prototype to start, and, as more answers are collected, the more the requirements for the prototype are refined.

In the end, all the questions end up being answered and it is possible to reach a definitive conclusion towards the thesis statement. The sub-conclusions are also interesting to analyze and are a contribute themselves to this area.

4.2 Approach

To answer these questions from Section 4.1.3, the work of this dissertation was divided into two main parts, each of them comprising a chapter:

1. **Supply Chain Issues Validation and Requirements Elicitation**
2. **Solution Design and Implementation**

Each of these focuses on reaching a conclusion towards the answers that underlie the thesis statement.

4.2.1 Supply Chain Issues Validation And Requirements Elicitation

The first question to answer in order to reach any conclusion is ***"What supply chain issues, improvements and requirements do the experts really find the most important?"***. As already mentioned, the other questions depend on the answers that we reach for this question.

The proposed way that was used to get an answer was an online **survey**. The survey serves the following purposes:

- Collecting quantitative data on the relevance of the issues that supply chain management suffers from.
- Collecting quantitative and qualitative data on which are the major points of improvement and compare their relative importance.
- Collecting quantitative and qualitative data on which use cases the experts think Blockchain can be more useful to accomplish.
- Collecting quantitative and qualitative data on which the functionalities that supply chain requires of information systems and blockchain.
- Correlate some of the data collected in the previous point and reach some extra conclusions that might help decide on the validity of the results.

The survey was designed to be answered by people with experience in the field of Supply Chain, with knowledge about Blockchain being optional but appreciated. These are the opinions that have the most importance on this field, and these are also the people that interact with the systems in question and can more accurately pinpoint the points of failure and improvement.

The survey was distributed over the internet, through relevant media and social forums, as well as through direct contact with professionals from the area. It was shared on some telegram channels of projects that apply Blockchain to the supply chain (most of which mentioned in the state of the art), on Reddit, through their supply chain and logistics forums, as well as on supply chain focused forum websites. It was also distributed to professionals through messages on LinkedIn (with a focus on managers), personally and through email.

In conclusion, the goal of the survey is to validate some assumptions and direct the dissertation towards the most important supply chain management problems.

4.2.2 Solution Design and Implementation

This section is focused on applying the knowledge of which are the most important aspects of supply chain to focus on improving, to answer the other two final questions: *"What is the Blockchain tool or framework most adequate to the development of an architecture that can support these requirements?"* and *"Is it possible to build a feasible architectural design, by using such a tool, to implement all these requirements?"*

The state of the art review documented in Chapters 2 and 3 already provided some information towards the answer to the first of these questions, through the comparison of frameworks. Therefore, and taking the results of the survey into account, a brief analysis is undertaken to make a decision.

Finally, and to answer the final question about the possibility and feasibility of building a blockchain-based architectural design for supply chain management, the implementation of a proof of concept is undertaken, using the chosen platform.

This approach is divided into the following phases:

1. **Requirement Elicitation and validation** - through the literature review research and survey results

2. **Design** - using the requirements, building a model that can implement the elicited requirements

3. **Implementation** - program the system according to the design

4. **Validation** - check if the built system satisfies the initial requirements fully, and if otherwise, explaining why not

After this approach is finished, the results are analyzed qualitatively, as to which use cases are implementable and usable or not and as to what were the limitations found because of the various decisions taken. Taking the various points of this analysis into consideration will allow for some conclusions to be taken in reference to the remaining question.

After having answered all of the three questions that underlie the thesis statement, it is possible to analyse the validity of the thesis statement ,though future work may eventually bring different answers, since Blockchain is a technology which is seeing great and fast developments, due to the amount of research being put into it.

Chapter 5

Supply Chain Issues Validation and Requirements Elicitation

The opinions of experts from a field can be highly valuable for research to achieve valuable contributions for that field. The optimal way to validate the issues of the supply chain would be to interview experts in a formal way. However, there were difficulties in this aspect, especially because most of the contacts did not go through and a lot of time and effort put into this was not fruitful. Therefore, since expert knowledge was still necessary, it was decided that a survey would be a more efficient method of gathering opinions and results.

The survey was written during the months of March and April, 2018, with results being collected from the end of April up until the mid of June. The purpose, methodology and results of the survey are analyzed in this chapter, leading to the validation of the issues mentioned in the previous chapters. The survey and analysis done here also serves as a basis for the requirements that were used to build a proof of concept prototype.

However, due to various factors, the survey and collection of results was delayed from the initial planning, resulting in the collection of results only being completed in June. Thus, the PoC used some assumptions from both the background analysis and the few initial answers from the survey, and kept iterating the requirements based on the new answers received during the development.

5.1 Purpose and Target

Even though the survey was delayed, the answers given are still important in the end. The main purpose of this survey, as stated in 4.2.1, was to give a response to the question we are focusing on: **"What supply chain issues, improvements and requirements do the experts really find the most important?"**.

This is done by dividing the survey into groups of questions: some about the issues of supply chain, some about the points of improvement as well as the opinions applicability of Blockchain as a system for supply chain management. Together, these allow for an opinion to be formed about the quantitative measurements of the importance of each issue, as well as a qualitative collection of requirements.

The survey is aimed at people with knowledge in the area of Supply Chains, and optionally Blockchain. The combination of knowledge in both areas is relevant for the questions that focus Blockchain applicability. However, having low knowledge about Blockchain is also relevant, so that the answers are more diversified for both sides, thus reducing the bias.

5.2 Methodology

This section describes how the data was collected and analyzed. It gives a brief overview of what questions were asked, how they were grouped, the objectives of each question, along with the type of answers used.

5.2.1 Data Collection

The data for the survey was collected by sharing the survey with professionals of the supply chain area, through contact with related companies, personal contacts, social networks like LinkedIn, Reddit and conversation groups.

The collected sample size was 25. Even though these are answers from knowledgeable people, the sample size is not very significant, so the margin of error for this study is higher than if it were otherwise.

The questions of the survey can be roughly grouped into four main groups of questions. Each of the groups has similar questions, with a distinct purpose.

1. **Question Group 1 - General Information**

- **Questions** - Information about the respondent's contact with supply chain, role in the industry, years of experience and number of employees in the current company (if in a job related to supply chain).
- **Goals** - Characterization of the respondent's, especially in order to validate the relevance of their profiles.

2. **Question Group 2 - Blockchain Knowledge and Opinions**

   - **Questions** - Questions about the general knowledge of Blockchain technology, opinions about Blockchain cryptocurrencies and their usefulness in various contexts, as well as Blockchain regulation and the effect of GDPR on Blockchain technology.
   - **Goals** - Get a vision on the need or not of cryptocurrencies for Blockchain related architectures, as well as the impact of regulations on this technology.

3. **Question Group 3 - Supply chain Points of Focus and Problems**

   - **Questions** - Questions to rate affirmations that deal with supply chain issues, focusing on the level of agreement that the participants have with the affirmations. Most of the affirmations correlate supply chain major issues with the points of failure that might underlie them; The others rate specific problems in an importance scale or by percentage of occurrence.
   - **Goals** - Validation of the issues that affect the supply chain management metrics the most, as per the opinion of specialists.

4. **Question Group 4 - Supply Chain Points of Improvement and Applicability of Blockchain Technology**

   - **Questions** - The first half of these questions serves to prioritize the importance of some supply chain issues, as well as the importance of some functionalities that supply chain information systems should feature. The second half of the questions ranks the use cases and benefits that Blockchain could bring to supply chain management.
   - **Goals** - This is the main point of the survey, and from here, the most important information towards the conclusions we want from the survey is gathered, which consists in the supply chain issues leading to Blockchain system requirements.

Most of the important questions use statements with the Likert scale in the answer, from 1 to 5, where 1 means "Strongly Disagree" and 5 means "Strongly Agree". In this survey the middle of the scale, 3, is "Neutral" or "Neither Agree nor Disagree" and there was a separate answer with "Do not know". Therefore we can classify this 1 to 5 scale as **ORDINAL**, since all of the values directly relate to a scale of increasing agreement. Additionally, some of the other questions also use a scale of importance from 1 to 5, while the remaining questions mostly have nominal answers (meaning that the group of answers for a question are fixed, qualitative and not numerical).

5.2.2 Data Analysis

The analysis of the answers is done through bar graphs and using mostly the **measures of central tendency or location**, **measures of central spread, scale or dispersion** and **measures of skewness and kurtosis**. The metrics used to analyze the data set and reach some conclusions are therefore: the mean, median and mode, the standard deviation and range, and the skew.

The meaning behind these metrics and how they are applied here can be found in annex A.1.

To classify if the skew of the distribution is high or not, there is a measure that can be calculated, the skewness standard error, with the following formula: $\sqrt{\frac{6*N*(N-1)}{(N-2)*(N+1)*(N+3))}}$, where $N = samplesize$.

For a sample size of 25, the Skewness Standard Error is 0,463683501.

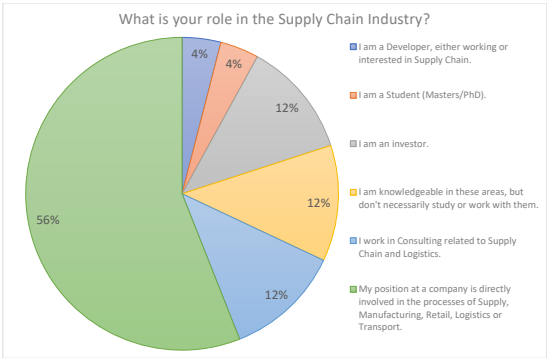
5.3 Results

The results of the survey are analyzed in this chapter [1]. Each group of questions features the graphs and metrics of the corresponding questions, and a brief informal analysis is done, based on the values.

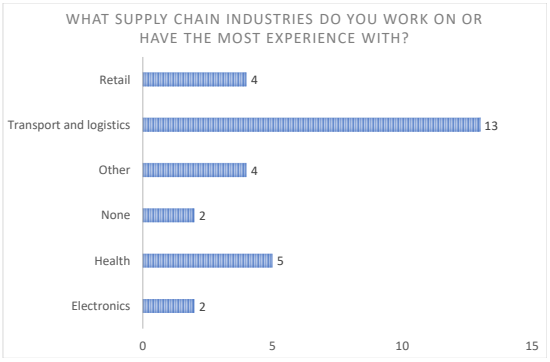
5.3.1 Question Group 1 - General Information and Participant Classification

Data from the participants was inquired, including: role, industry, years of experience and company size (if they worked in one at all).

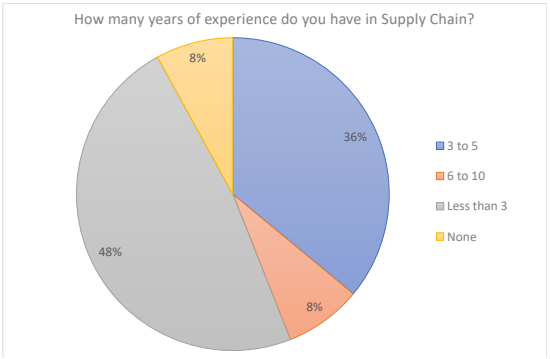
[1] The results are available at: https://drive.google.com/file/d/1Zyg8BGj0dV6KFKPZdshgbYHYi-m8snGX/view?usp=sharing



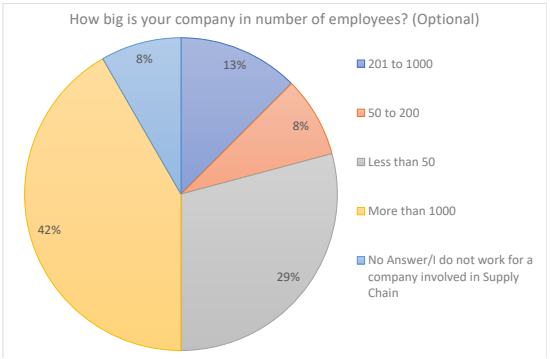
(a) Role played in relation to Supply Chain Management



(b) Industry



(c) Years of experience



(d) Number of employees in company

Figure 5.1: Questions about the respondent's Role, Industry, Years of Experience and Company Size.

The data, as can be seen in Figure 5.1, can be summarized in the following statements:

- Most people, about **68% of the respondents, have an active role working in areas related to supply chain**. From these, 56% work in a company directly involved in the processes of supply, manufacturing, logistics, retail or transports and 12% work in consulting related to these areas. The rest are people with a profile that gives them knowledge about supply chain, be it through education or other type of contact with the field.

- **The most common area of supply chain to work in is, by far, Transport and Logistics**, with 52% of the respondents saying they have work experience in this field. **Health and Retail** are also common areas, with 20% and 16% respectively.

- 48% of the respondents have less than 3 years of experience, with 36% having 3 to 5 years and 8% having 5 to 10 years. However, in total, there were **only 2 respondents (8%) without any years of experience in the field**.

- 62,5% of the respondents work in medium to big companies (more than 50 workers), from which 41,7% work in companies with more than 1000 people and 12,5% in companies with between 200 and 1000 people.

Conclusions from Question Group 1

From this data, we can ascertain the general profile of the survey respondents. These are people who, in their majority, work in big companies related to the supply chain, in the various fields, but mostly in transport, logistics and health. The respondents, though they may have had education in this area, do not show a high number of years of experience, with most of them having only up until 5 years of experience.

5.3.2 Question Group 2 - Blockchain Knowledge and Opinions

This section features some questions that inquire about the respondent's knowledge in blockchain, and their opinions about the relationship between cryptocurrencies and blockchain. Are blockchains independent of cryptocurrencies or not? And in which cases? These are questions that will also help decide the need of a cryptocurrency for the design of the system that was proposed.

1 - General Blockchain Knowledge.

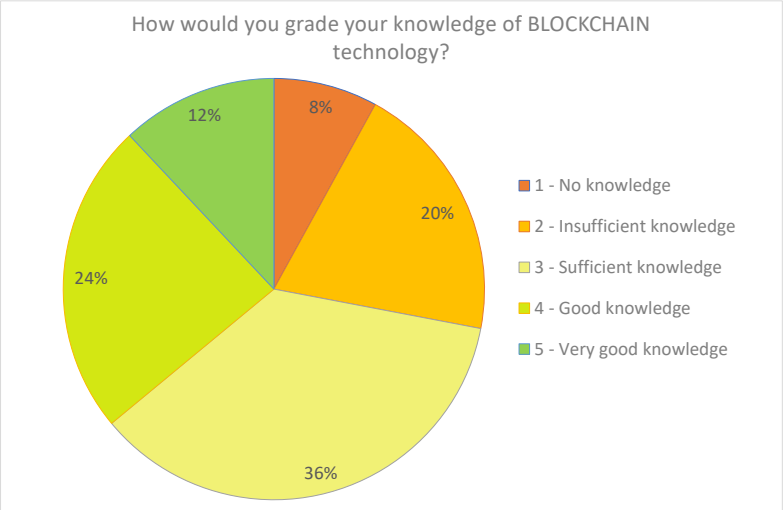


Figure 5.2: Question about Blockchain knowledge.

In Figure 5.2, it can be seen that most answers are in the middle of the scale, while both the extremes have few answers, which closely resemble a gaussian or normal distribution.

Indeed, when calculated, the skew of this data is about -0,065104294, which is a very low number (the standard error was about 0,46) and close to 0 (zero). A normal distribution has a skew of 0 (zero), so this can be said to be a good approximation. Additional, it can be said that there is not much of a bias towards accepting or denying blockchain when answering other questions, either from knowing too much or too little, since the knowledge curve seems to be normal.

2 - Rate the affirmations about the use of cryptocurrencies in the blockchain.

After inquiring about the blockchain knowledge of the respondents, some affirmations were made about cryptocurrencies, which can be seen below. The question was optional, since respondents without sufficient knowledge of blockchain might face difficulties understanding the affirmations.

Affirmations:

1. A blockchain always needs to have a cryptocurrency attached in order to work well.

2. Cryptocurrencies are useful in blockchains open to the public, in order to create incentives for good behavior, but not always necessary in private chains.

3. Blockchains can be useful in some uses cases, independently of whether they hold a cryptocurrency or not.

Independently, these affirmations do not hold an absolute value that can help ascertain if there is a need for a cryptocurrency in the case of supply chain. But together, the three affirmations lead to a higher degree of certainty.

The first affirmation states that a blockchain, any at all, independently of the purpose or type, does indeed need a cryptocurrency. The second affirmation goes on a tangent from the first one, stating that maybe cryptocurrencies are only needed as an incentive, and otherwise could be not needed. The final affirmation is more generic in nature, and attempts to **completely separate the concept of blockchain and cryptocurrency**. The results for each affirmation can be found in Figure 5.3 and in Table 5.1.

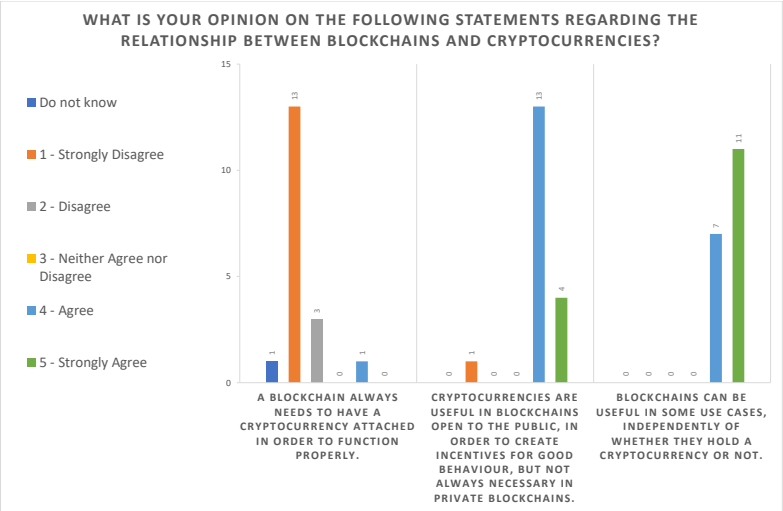


Figure 5.3: Question to rate affirmations about the use of cryptocurrencies in the blockchain.

Table 5.1: Question results metrics for the affirmations about the use of cryptocurrencies in the blockchain.

	Mode	Median	Mean	Standard Deviation	Range	Skewness
1. Blockchain needs cryptocurrencies to work well	1	1	1,36	0,79	3	2,74
2. Cryptocurrencies are only useful as incentives in public blockchains	4	4	4,06	0,87	4	-2,51
3. Blockchains can be useful in some cases, regardless of having or not cryptocurrencies	5	5	4,61	0,50	1	-0,50

- The first affirmation has a very low agreement rate. Almost all of the respondents answered that they strongly disagreed with it, which can be seen in the table, as both the mode and median are 1, and the mean is also close to it. It can be concluded that, according to these opinions, **blockchain can sometimes not have a cryptocurrency and still perform well**.

- The second affirmation has a high agreement rate, with an average of 4.06. The mode and mean are also very close to the mean, and the standard deviation is lower than 1, which means that there is a consensus on the responses. Therefore, it can be concluded that, as per the opinions of the professionals, **a cryptocurrency is more useful in public blockchain contexts and applications, as an incentive, than in private blockchains, where these incentives are not needed.**

- The last affirmation has a very high agreement rate, the highest of them, with a mode and median of 5, and a mean of 4.61. There is a strong consensus, as all of the respondents answered with either 4 or 5 (Agree or Strongly Agree), leading to a low standard deviation and range. There is a slight skew, since the answers are distributed between these 2 items, with a bigger number of responses on item 5. The conclusion for this affirmation is that **blockchains are useful, independently of holding a cryptocurrency or not**.

Conclusions from Question Group 2

With these conclusions in mind, one final conclusion can be derived. If blockchains do not need cryptocurrencies to perform well (concluded from the non agreement with affirmation 1) and can generally still be useful without them (affirmation 3), then **cryptocurrencies should not be considered a necessity, but rather a design option, depending on the benefits** it can bring to a particular use case. Taking this conclusion into account, and also according to the agreement with affirmation 2, then: **if a blockchain design features a private blockchain, it does not necessarily need a cryptocurrency as a requirement, since incentives are not needed**. This is an especially interesting conclusion for the proof of concept.

5.3.3 Question Group 3 - Supply Chain Points of Focus and Problems

This part of the survey focuses on finding out if the issues dug out on the background and pointed out in the Problem Statement chapter are really the ones that plague supply chain management. As such, a battery of questions including the various concerns mentioned was given to the respondents.

1 - Speed of Delivery, synchronization and traceability issues

The respondents were asked the following 2 questions, as well as some affirmations to classify their agreement with.

Questions:

1. In your experience, how frequently does the absence of crucial information cause delays in processes such as shipping, packaging, etc.?

2. How important is Inventory Management for the efficiency of the supply chain?

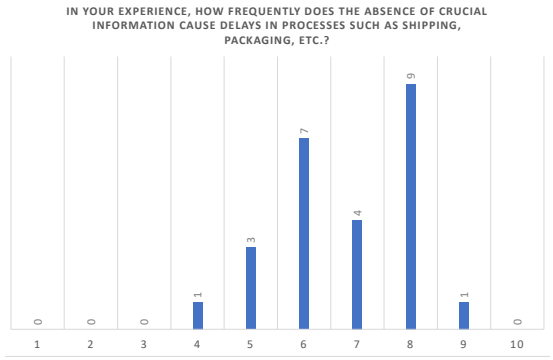
Affirmations:

1. In a supply chain, if the information gap between partners is shortened, a faster product cycle (or shorter lead time) can be attained.

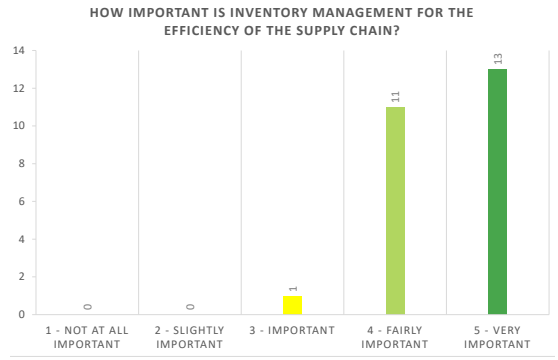
2. Distribution, supply, demand and inventory planning all rely heavily on the information being both accurate and up to date

3. Supply Chain Management lacks a way to quickly and seamlessly share between companies all the information generated by the flow of assets in a Supply Chain.

The data collected from the first 2 questions can be visualized in the graphs from Figure 5.4. The data for the 3 affirmations can be visualized in the graphs from Figure 5.5 and was summarized in Table 5.2.

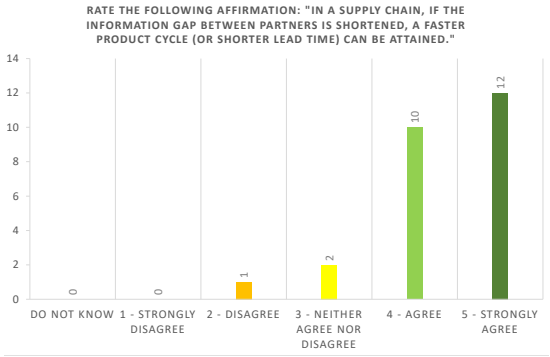


(a) Inventory Management Importance

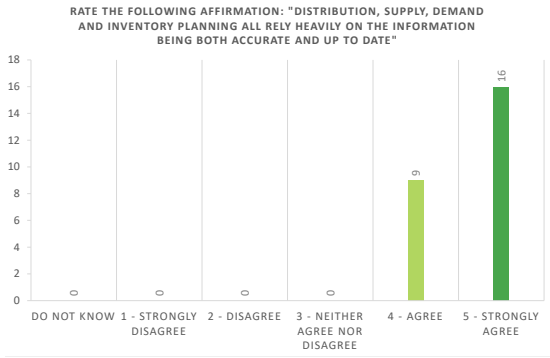


(b) Relation between the absence of information and delays

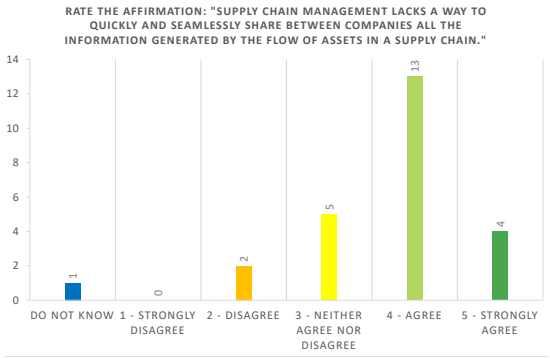
Figure 5.4: Questions to rank inventory management importance and relationship between absence of information and delays.



(a) Relation between information gap and delays



(b) Relation between management planning and information accuracy



(c) Affirmation about the lack of a system with good integration of information

Figure 5.5: Questions to rate affirmations about the effects of information gaps, the reliability of management planning and lack of a system with good integration.

Beginning with the analysis of the 2 questions:

- The first question had the objective of finding out whether Inventory Management is an important discipline for the efficiency of the supply chain, which is an important aspect, if it is to be treated as point of possible improvement. The data was ranked from 1 to 5, with 1 being "Not at all important" and 5 being "Very Important". The question scored a mean of 4.48, with the mode and median both being 5, since most results sit on that score. There is little dispersion in the results, with most of the other answers scoring a 4 and only 1 answer scoring a 3. It can be concluded that **Inventory Management is an essential discipline for supply chain management, according to the professionals.**

- The second question tries to relate the absence of information with delays in the processes of sending products. The scale went from 0 to 10, which corresponded from the percentages 0 to 100%. The final results ranged from 40% to 90%, with most of the answers sitting between 60% and 80%. The results were not very spread out, with a standard deviation of only 1.29, in the scale of 1 to 10. There

Table 5.2: Question results metrics for the affirmations about the effects of information gaps, the reliability of management planning and lack of a system with good integration.

	Mode	Median	Mean	Standard Deviation	Range	Skewness
Affirmation 1 - A reduction of the information gap between partners can lead to a faster product cycle.	5	4	3,92	1,26	3	-1,21
Affirmation 2 - Distribution, supply, demand and inventory planning all rely heavily on the information being both accurate and up to date.	5	5	4,64	0,49	1	-0,62
Affirmation 3 - Supply Chain Management lacks a way to quickly share between companies all the information from the supply chain.	4	4	3,79	0,83	3	-0,56

were not also any significant outlier values, with the skew being classified as normal, with a value below the threshold of the standard error. **The average result for this opinion was 68%, which is a reasonably high percentage that we can say there might really be correlation between these 2 aspects, especially given the consensus and non-existence of outlier values. However, this can never be said with total certainty, since these are only opinions and not factual data**.

A follow-up with the analysis of the opinions about the affirmations:

- The first affirmation related a reduction in the information gap with a faster product cycle. In layman's terms, this can be thought of as "if there is more information available, the products will get made faster, shipped faster and received faster". Many respondents seemed to only cautiously agree with the affirmation, not committing to a strong answer. Even though the most popular answer was 5, the mean still was below the normal agreement level, at 3.92. There is some skew and spread, due to the lower results, but these still look like valid answers. Therefore, it can be informally generalized that **while it may be true indeed that having information available will generally help products finish their cycle faster, it might not always be the case, as there is not a strong consensus about this.**

- The second affirmation relates all the planning disciplines in SCM with the existence of up-to-date and accurate information. The respondents all answered

with either 4 or 5, a very low spread of answers, with most of them strongly agreeing (agreement average of 4.64. It can then be concluded that having accurate and up-to-date information is essential for these planning management tasks, which, of course, affect inventory management, which we already concluded to also be essential. **It is, therefore, of the utmost importance for a supply chain to be provided timely with exact information**.

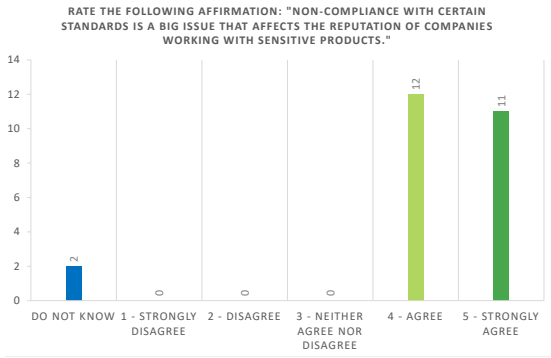
- The last affirmation states that supply chain management might lack a way to integrate the information provenient from the flow of assets between companies easily. However, out of the 3 affirmations, this is the one with the lowest agreement of them. While the average agreement, 3.79, is close to the one from the first affirmation, the most popular choice here was 4 (Agree), followed by 3 (Neutral). The skew is almost low enough that this would represent a normal distribution for the answer. **While it seems that not every professional agrees, and some think that there are already good ways to integrate information quickly and seamlessly, the majority still thinks that SCM is lacking on this aspect.**

2 - Quality assurance and traceability issues

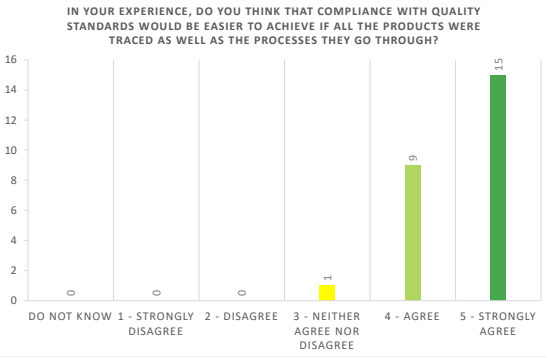
Finally, with the objective of ascertaining whether quality assurance is also an issue, the respondents were asked 2 questions, with the results being shown in Figure 5.6. The first, similarly to previously, was asking about the agreement with an affirmation, and the second was a question. Both the affirmation and question are as follows:

1. Non-compliance with certain standards is a big issue that affects the reputation of companies working with sensitive products.

2. In your experience, do you think that compliance with quality standards would be easier to achieve if all the products were traced as well as the processes they go through?



(a) Relation between standard non-compliance and company reputation



(b) Relation between process traceability and quality standards compliance

Figure 5.6: Questions about quality assurance issues.

Looking at the graphics, the second question has a slightly higher agreement rate than the first one. However, both questions have very high agreement results, with absolutely no disagreement answers and only 1 neutral answer on the second question. It can be concluded that **quality standards compliance might affect a company's reputation and is also somewhat dependent on the traceability of products and processes they go through**.

Conclusions from Question Group 3

- From the first set of questions, we summarily conclude that the accurate synchronization of data and the speed of delivery (metrics stated in the problem statement) are related and an improvement in the synchronization might positively affect the speed of delivery, as well as other important metrics of a product's cycle. Though there may be solutions for this, the professionals still that they might be lacking.

- From the second set of questions, the only conclusion was that companies have in their interest to follow the quality standards, and traceability plays a big role in this.

The questions from this group have successfully validated most, if not all, of the supply chain issues raised in the previous chapters. The remaining questions from the survey point towards points of improvement for these issues, as well as what possible features to implement in a system could help with these issues.

5.3.4 Question Group 4 - Supply Chain Standalone Points of Improvement and Blockchain points of Applicability and Improvement for Supply Chain

The first goal for this part of the survey is to find out the points of improvement for supply chain. What is being questioned are not the issues of the supply chain themselves, which were already asked, but what parts of supply chain, even though they may already work well, could work even better.

The second goal is to find out more about the requirements that an information system must have to satisfy the needs of SCM. This includes ranking the functionalities of an information system for their importance. Many of these functionalities probably relate to the points of improvement found here, as well as to the supply chain issues described in Section 5.3.3.

For the sake of simplifying the ranking process for both the *"points of improvement"* and *"functionalities of an information system"*, a division into groups of importance shall be done according to the observed results. The importance scale went from 1 to 5, but most of the mean scores were between 3 and 4.5.

Therefore, the following classification for the lists of results will be adopted:

- **Mean** $\geq$ **4** - High priority or importance
- **3.5** $\leq$ **Mean** $<$ **4** - Medium importance
- **Mean** $<$ **3.5** - Low importance

1 - Rank the importance of some aspects which Supply Chains aim to improve.

In this category, we can see the results on Figure 5.7 and then represented on Table 5.3. Pretty much all the options had a range of 3 and deviations close to 1, which is to be expected. Traceability had the highest skew value, as there seemed to be some dissent, as a few outlier values pushed the mean to be lower than the median. However, it still rated the highest, by importance.

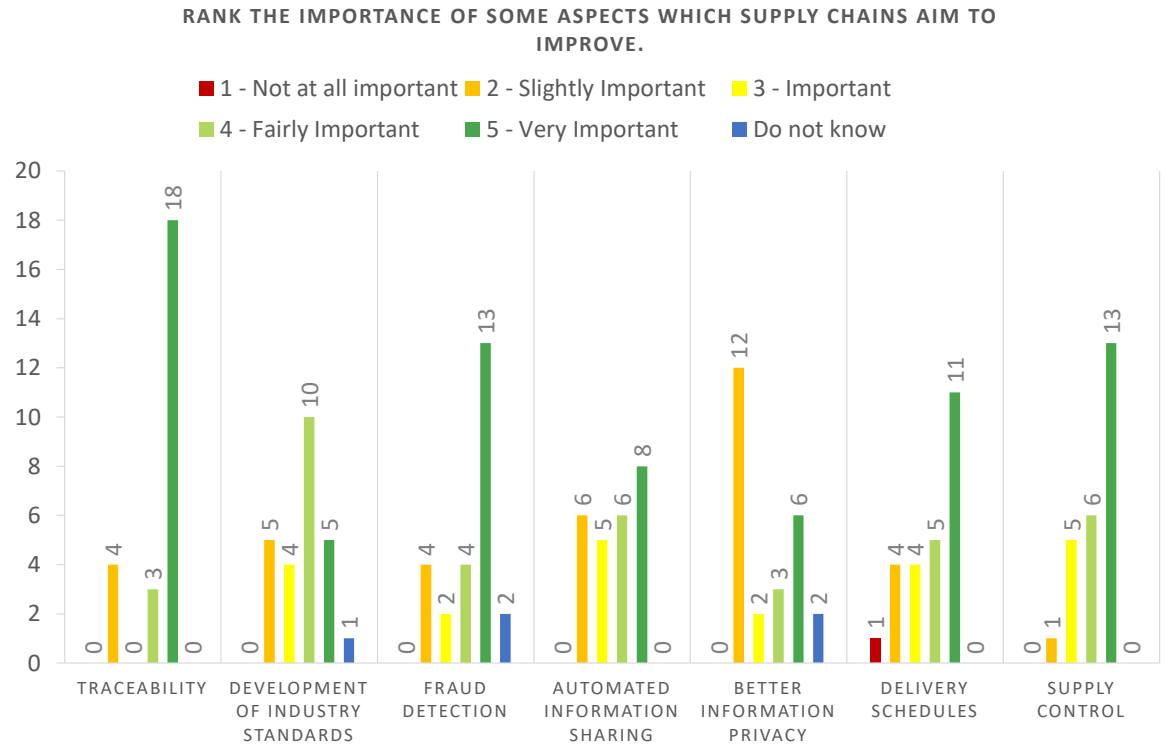


Figure 5.7: Question: "Rank the importance of some aspects which Supply Chains aim to improve."

Table 5.3: Question results metrics for the question to rank the importance of improvement aspects of the supply chain.

	Mode	Median	Mean	Standard Deviation	Range	Skewness
Traceability	5	5	4,40	1,12	3	-1,67
Development of Industry Standards	4	4	3,63	1,06	3	-0,36
Fraud Detection	5	5	4,13	1,18	3	-1,00
Automated Information Sharing	5	4	3,64	1,19	3	-0,20
Better Information privacy	2	2	3,13	1,32	3	0,51
Delivery Schedules	5	4	3,84	1,28	4	-0,71
Supply Control	5	5	4,24	0,926	3	-0,87

The importance ranking described as before follows below:

1. **High importance** - "Traceability", "Supply Control", "Fraud Detection";
2. **Medium importance** - "Delivery Schedules", "Automated Information Sharing", "Development of Industry Standards"
3. **Low importance** - "Better Information Privacy"

Conclusions:

- High Importance - It had already been concluded from the previous set of questions in 5.3.3 that inventory management was an essential discipline to SCM, so it was expected that *Supply Control* would be rated highly as a point of improvement, which it did. The same can be said about *Traceability*: it was expected to rank highly, since it is related to both quality assurance and availability of information, two other issues that were validated in the previous set of questions. At the same time, *Fraud Detection* was also highly ranked, which can be explained by its relation to traceability and quality compliance and control. **Traceability helps to avoid fraud and, at the same time, fraud detection also relates to quality control and compliance, since avoiding fraud is all about making sure that the products circulating the supply chain are authentic and have good quality.** With this in mind, it is only natural that Fraud Detection is considered important, together with traceability.

  In conclusion, the 3 most important items are traceability, supply control and fraud detection, with supply control and fraud detection directly depending on the traceability, which makes traceability probably the most important point of improvement in the supply chain.

- Medium Importance - From the items that got rated as medium importance, 2 of them relate to synchronization improvement: *Automated Information Sharing* and *Development of Industry Standard*. The last point, *Delivery Schedules* also relates to Supply Control, though it is less generic.

  In conclusion, though system synchronization is not as important as traceability or some of the other items that depend on traceability, it is still a welcome improvement which comes in second place to the others.

- Low Importance - Finally, the lowest ranking item was *Better Information Privacy*. This comes as a surprise, since security was a big focus on the background research for supply chain management.

  The conclusion we can take from this item being rated the lowest, is that it might be a concern that the current systems already take care of, therefore it does not have much space for improvement.

2 - Rank the importance of the following functionalities for an information system that stores or processes data pertaining to a Supply Chain

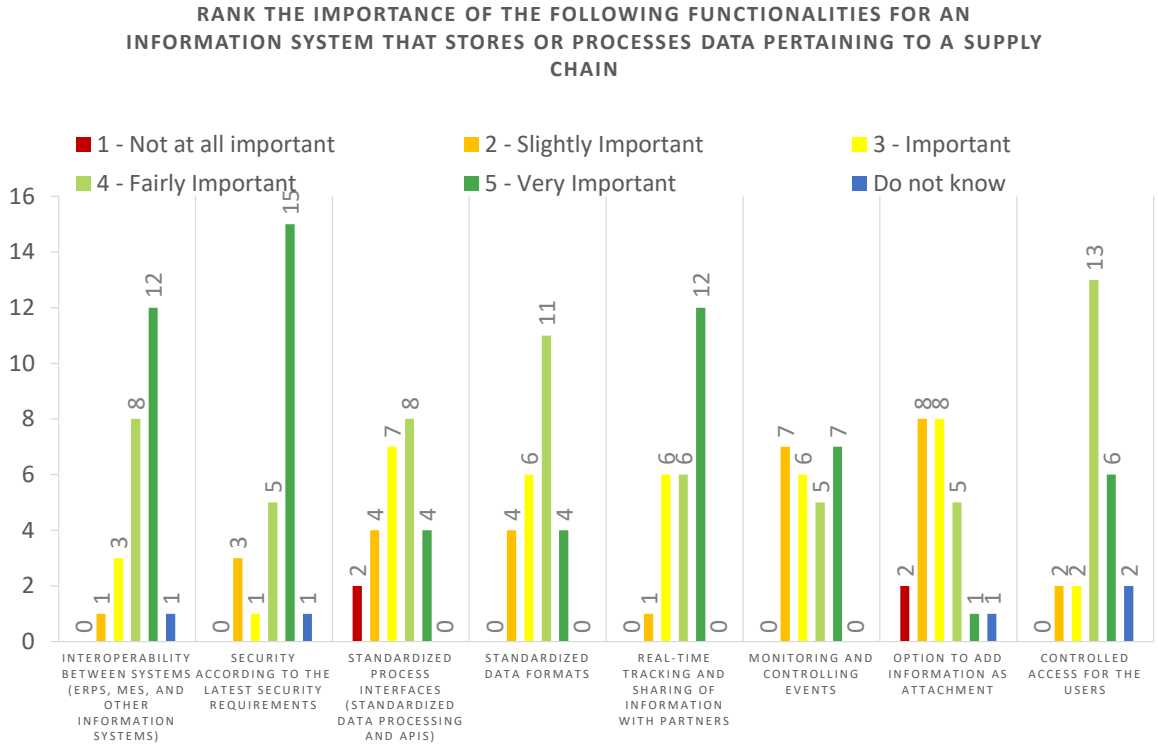


Figure 5.8: Question: "Rank the importance of some functionalities in a supply chain based information system."

Results for this question are found in Figure 5.8, and the analysis metrics are in Table 5.4 the range of answers is pretty much 3 for all options but 2 of them. The mean and median of the items are also a close approximation of each other, which happens because the skew in most cases is not very high (only for the Interoperability and Security items was it higher than 1). The skew is highest on the items that also had the highest mean. This happens because, even though these items are the most

Table 5.4: Question results metrics for the question to rank the importance of functionalities in an information system for supply chain.

	Mode	Median	Mean	Standard Deviation	Range	Skewness
Interoperability Between Systems (ERPs, Manufacturing Execution Systems, and other information systems)	5	4,5	4,29	0,86	3	-1,08
Security according to the latest security requirements	5	5	4,33	1,05	3	-1,49
Standardized Process Interfaces (Data Processing and APIs)	4	3	3,32	1,18	4	-0,36
Standardized Data Formats	4	4	3,60	0,96	3	-0,31
Real-time tracking and sharing of information with partners	5	4	4,16	0,94	3	-0,67
Monitoring and controlling events	5	3	3,48	1,19	3	0,05
Option to add information as attachment	2	3	2,79	1,02	4	0,19
Controlled access for the users	4	4	4	0,85	3	-0,96

popular, there are always a few answers from people who disagree. However, this is to be ignored, given the low deviation and given that the median and mean are very close to each other in every case. Therefore, the mean can still be used here to classify the importance.

The importance ranking described as before follows below:

1. **High importance** - "Security according to the latest requirements", "Interoperability Between Systems", "Real-time tracking and sharing of information", "Controlled access for the users";

2. **Medium importance** - "Standardized Data Formats"

3. **Low importance** - "Monitoring and controlling events", "Standardized Process Interfaces", "Option to add information as attachment"

Conclusions:

- 2 of the high importance items actually relate to security, namely access control and following security requirements. This contrasts with the point of improvement from the previous set of questions *"Better information privacy"*, which was rated the lowest in relative importance. There seems to be a disparity between the importance of the functionality and the importance of the improvement, which might mean that the functionality is important but already good enough as it is.

The conclusion we can take is that keeping security up to date still seems to be considered important, but on aspects other than privacy (which is already fulfilled) such as access control, which relates to authorization and authentication.

- The other 2 high importance functionalities were about interoperability between systems, and tracking and sharing information. Tracking and sharing information relates to traceability and information synchronization, while interoperability relates only to synchronization.

  Though these points, which relate to synchronization scored high as a functionality, the points of improvement they relate to only scored as medium in the previous sections, namely *"Automated Information Sharing"* and *"Development of Industry Standards"*. Thus, integrating the information is seen as an important aspect to have,but not one that needs improvement over the current status.

- Most of the items rated as having medium or low importance do not necessarily stand out nor do they have important conclusions to be taken. The one thing that stands out is that *"Monitoring and controlling events"* rated as a low importance functionality, even though traceability rated high as a point of improvement, which does not make a lot of sense, since traceability also means having the means to monitor what is happening.

3 - Blockchain applicability to the supply chain

The final important group of questions featured 2 questions about the viability of applying blockchain to the supply chain. The results are shown in images 5.9 and 5.10 The first question asked people which use cases they thought were the best for applying blockchain to the supply chain and respondents could select a maximum of 3 options (including "None" or "Do not know").

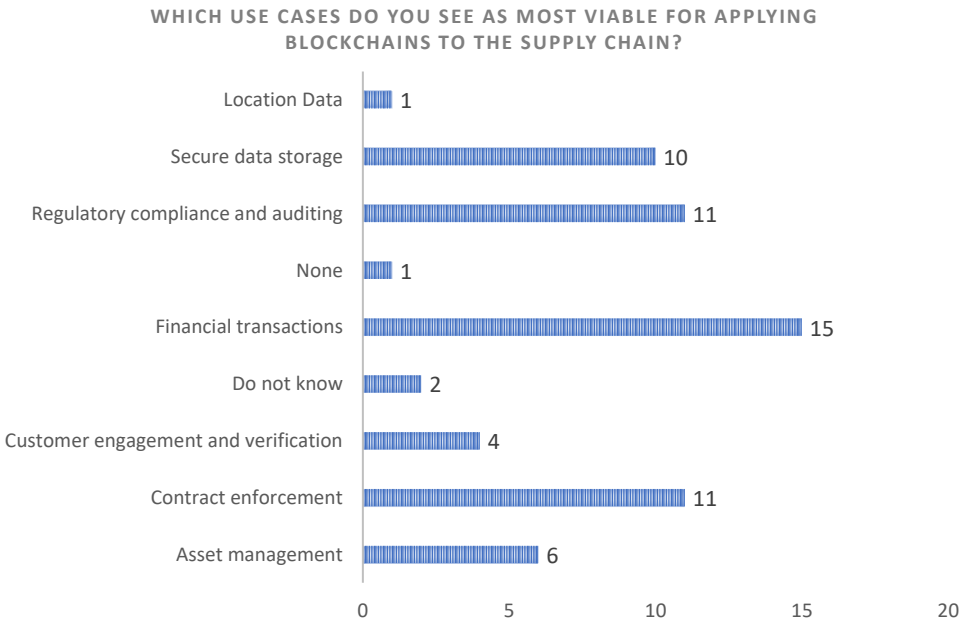


Figure 5.9: Question about the Blockchain use cases for the supply chain.

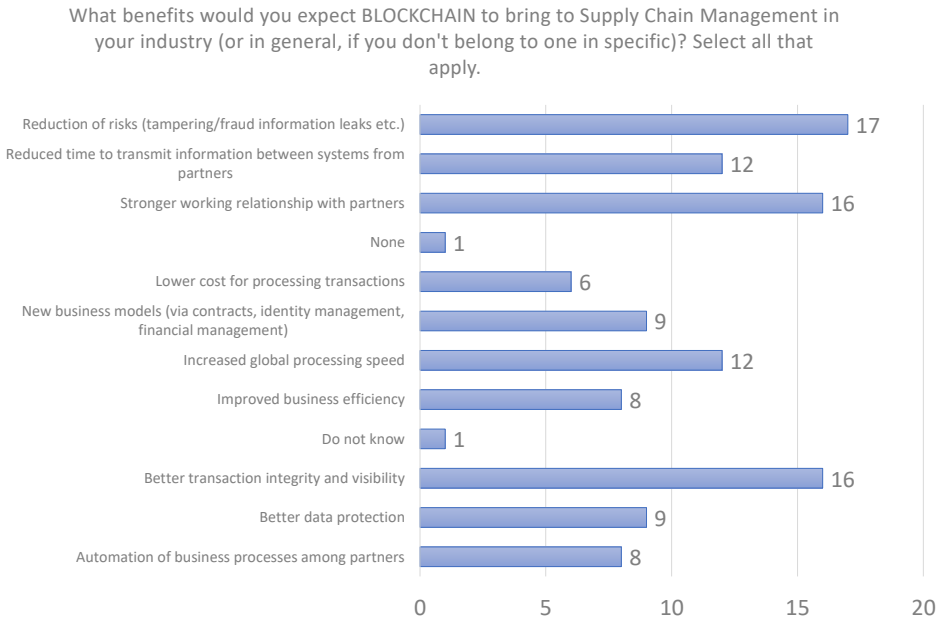


Figure 5.10: Question about the Blockchain benefits for the supply chain.

Starting with the analysis of the first question:

- **Financial transactions was, by far, the most popular choice.** The respondents look at blockchain as a way to move money as an asset, more than a tool to manage the goods themselves, since traditional systems only deal with data and the money is dealt with by third parties (banks). It makes sense that **the second most voted choice is Contract Enforcement.** As per the background research

and state of the art, it was pointed out that some of the applications already enforce payments through contracts, when the goods are delivered,for instance, without the need of a third party (thus reducing fees). Financial transactions and contract management go hand in hand.

- With the same number of votes, **also in second place, is "regulatory compliance and auditing"**. Regulatory compliance is a term also related to contracts enforcement. Contracts are used to establish mandatory rules which are automatically enforced according to the conditions of the system (for instance, enforce that the food transported in a truck never goes over a certain temperature value). **However, an important point to note is that regulatory compliance and auditing are made possible through the traceability characteristic of blockchain, which allows for products and processes to be thoroughly verified. This means that traceability is a highly sought after application of blockchain.**

- "Secure data storage" was also a highly voted option. Security is an important focus in information systems, so this should be taken into account when building the proof of concept. In the previous set of questions, "access control" and "security according to the latest requirements" were also features that stood out, so **it is not a surprise that secure data storage is seen as one of the main advantages of blockchain, since it boasts of being cryptographically secure, having immutability by design and an easy to implement access control.**.

The proof of concept has to not only take these use cases into account as the main applications, but also implement (or make possible the implementation of) the most important features of information systems previously analyzed, which makes the second question complementary of the first one.

For the second question, there is an analysis of the benefits that blockchain can bring to the supply chain, according to the respondents:

- The three most voted items were: *"Better transaction integrity and visibility"*, *"Reduction of risks (fraud/tampering/information leaks)"* and *"Stronger working relationships with partners"*.

  The first two items were expected to rank high. They directly relate to the security and traceability, points of improvement already discussed - traceability makes it possible to view the records to make sure nothing suspicious is going on, and the security and immutability of the information makes it impossible to tamper with the data.

The third point, "Stronger working relationships with partners", however, is interesting to see being chosen a high ranked benefit. The concept of strong relationships with partners is linked to the concept of **internal supply chain trust**, which was already introduced in 3.1. The ability for partners to trust each other is what makes the supply chain possible and efficient to the standards of today. It seems that the professionals really do think of blockchain as a tool to improve this internal trust.

The selection of these 3 items as the most voted does not seem to be a coincidence. **Blockchain can be used as a source of truth for the supply chain and this leads to the side benefits of internal trust, better relationships with the partners, less fraud, less tampering, less leaks, among other benefits, all made possible because of the main characteristics: security and traceability.**

Conclusions from Question Group 4

This was the most important group of questions in that it provides direct information as to which functionalities the proposed system design should possess and implement. At the same time, these functionalities can be related to the points of improvement found, as well as to the issues validated in the previous groups of questions.

- From question group 3, some issues in the supply chain were validated: **inventory management**, **quality assurance** and **getting accurate and timely data**. On this group, the main points of improvement were found to focus on: **traceability**, **supply control**, **fraud detection** and **synchronization**. In a generalized way, the common links to both issues and points of improvement seem to be: **security**, **traceability** and **synchronization**, effectively validating the initial concerns from the problem statement.

- Another important conclusion to take from this set of questions are the features to include in the design for the blockchain system. These include the ***"Blockchain only features"***, functionalities implemented in an unique to the blockchain, as well as the standard ***"Information system features"***, which are the features that were found to be the most important to information systems in a supply chain.

- By order of importance, the blockchain features are:

  - Financial transactions.
  - Some form of regulatory auditing.

  - Enforceable contracts (smart contract functionality).
  - Secure data storage.
  - Asset management.

- By order of importance, the most important information system requirements are:
  - Security according to the latest requirements.
  - Interoperability between systems.
  - Real-time tracking and sharing of information with partners.
  - Controlled access for the users.

5.4 Comparison to the state-of-the-art projects

Most of the projects analyzed in the state-of-the-art, thought they use a multitude of different frameworks and custom networks, also do not focus on satisfying all of the requirements, but mostly a subset of them. Most of the projects specialize only in 1 problem area, like traceability, instead of going for an all-rounded concept.

CargoX This project focuses somewhat on Secure Data Storage, Enforceable Contracts and Security, even though in a limited way, since the data stored is not for any asset or document, but specifically for bill of lading documents. Additionally, it lacks a lot in the traceability items.

Eximchain It attempts to tackle all of the areas: Traceability, Security, Synchronization, Contracts and Finance. Though it promises a lot in its presentations, the actual whitepaper and technical documents focus mostly on the financial and contractual aspects and do not explain how the other aspects will be achieved in detail. The product looks promising for this area, by focusing all the requirements in a way that is claimed to be efficient, with custom networks and consensus algorithms and protocols. However, and very importantly, Eximchain's product has not yet been proven to work. The project has 3 proof of concepts planned in their roadmap, each focusing on separate areas, but none has been presented yet.

OriginTrail This project boasts of its strong traceability capabilities, mainly for asset management, auditing and fraud detection. It is a specific project, but it does not fit all of the requirements above, as it misses the financial transactions and some security aspects, such as access control. Again, as is the case with Eximchain, the

project seems promising, but is yet to prove most of its functionalities, having released proof of concepts mostly for the traceability requirements. This seems to be a common element to most of these projects, in that they look promising but take a long time to prove that they can deliver their promises.

Ambrosus Focuses a lot on auditing, fraud detection, quality control and traceability in general, so it fills all the traceability requirements. Even though these features are a must, and seemingly the most important, most of the other requirements are not met, especially the financial transactions, asset management and part of the security requirements.

In summary, the existing projects are either very specialized and do not meet all the requirements, or they are very broad in the requirements, but in early stages of development, without proving that they can do what they promise.

5.5 Conclusions

Having analyzed all the groups of questions from the survey, it should now be possible to have an answer for the first question of the thesis sub-statements, presented in the problem statement: *"What supply chain issues, improvements and requirements do the experts really find the most important?"*.

Most of the conclusions for the survey were summarized in the analysis of question group 4. These conclusions included the most important issues of the supply chain, points of improvement, blockchain features and information system functionalities important to the design.

Issues - These include **inventory management**, **quality assurance**, and **lack of accurate and timely data**. This last item was confirmed by the respondents to have a correlation to the lack of good integration and synchronization tools and to be one of the possible causes for the difficulties in planning management and product cycle delays.

Point of improvement - These include **traceability**, **supply control**, **fraud detection** and **synchronization**. These items reflect themselves on the feature requirements that were elicited. The results can therefore be crossed with the points of improvement to make more refined logical groups of requirements:

Important Features - The most important features, gathered from a junction of information system features and blockchain features, are summarized in Table 5.5.

These are important contributions from the experts of the area, though with a

Table 5.5: Elicited requirements grouped by improvement area of focus.

Area	**Requirements**
Security	Security according to the latest requirements
	Controlled access for the users
	Secure data storage
Traceability	Regulatory auditing
	Fraud detection
	Asset management
	Real-time tracking information
Synchronization	Interoperability between systems
	Development of industry standards
	Real-time sharing of information with partners, leading to better working relationships
Transaction Enforcement and Financial domain	Financial transactions
	Enforceable contracts (smart contract functionality)

bigger sample, the answers would have been more representative. The next chapter will focus on these requirements as the major points of focus for the design.

Chapter 6

Solution Design and Implementation

Following the conclusions and requirements elicited from the previous chapter, it is now time to answer the remaining two questions. The first question, **"What is the Blockchain tool or framework most adequate to the development of an architecture that can support these requirements?"** requires an analysis of the frameworks. The second question, **"Is it possible to build a feasible architectural design, by using such a tool, to implement all these requirements?"** requires that the elicited requirements be formed into a list and that an architectural design be built and implemented using the chosen framework. Only at the end of these tasks can we verify if all the requirements are implemented by the PoC.

This chapter deals with explaining these tasks sequentially. The first section analyzes which frameworks best satisfy the requirements and a choice is made. The second through fourth sections resemble a software engineering approach: starting in the requirements specification, following it up with the design, implementation using the framework and finally, the validation of the requirements.

6.1 Framework Comparison and Choice

Following the conclusions and requirements elicited in the previous chapter, it is now time to answer the second question from the problem statement: **What is the Blockchain tool or framework most adequate to the development of an architecture that can support these requirements?**

In order to make a good choice for the framework, there is a need to make a mixed analysis that focuses on the most important functionalities that an information system

should feature as well as what use cases are most viable for applying blockchain to the supply chain.

In addition to the requirements from the previous chapter, the performance analysis from Chapter 2.4.5 should also be taken into account.

6.1.1 Framework Requirements Elicited from the Survey

The requirements for the choice of a framework can be partially derived from the most important functionalities pointed out on the results of the survey, in Table 5.5. These functionalities are directly based on some points of focus from this thesis: **synchronization, security and tracking or traceability**, so it is based on these attributes that the framework should also be selected.

Security: From Section 5.3.4, it was ascerted that the improvement of information privacy in supply chains was not very important in itself. Maybe it is something that the current systems already do well, but as we can see, **there are still some outstanding concerns which deal with other security aspects that still need to be taken care of**: access control and fraud detection seem to rank high on the requirements for such a system. **Thus, the selected framework should support a highly controlled environment, where the actions each user takes should be properly authorized and there are good authentication mechanisms. Some fraud verification checks should be supported**.

Traceability: For the traceability requirements, the proposed system should be able to track all information, including changes in the system, registries of assets, transactions, network participants and organizations. It should also be open to outside regulatory entities, so that they can look into the system for auditing. Additionally, sanity checks and other fraud-detection operations should be possible, possibly through smart-contract functionality. **Thus, the selected framework must support the management of data in the form of assets, entities and organizations, which should be accessible only to specific entities, including auditors**.

Synchronization: As the source of truth, the blockchain should be easily accessible to any external system that needs to query or insert data. The elicited requirements for synchronization include the development of standards and system interoperability for real-time sharing of the information. **Thus, the selected framework should easily expose the blockchain information to outside systems (for instance, through REST APIs) using predefined data formats**.

Transaction Enforcement and Financial domain: There is interest in financial ap-

Table 6.1: Summary of the framework requirements for each improvement area.

Area	Framework Requirements
Security	Highly controlled environment
	Authentication and authorization mechanisms
	Fraud verifications (by smart contracts)
Traceability	Management of data: assets, entities, organizations
	Data access controlled, but accessible to auditors
Synchronization	Expose data to the outside systems through APIs
	Allow the use of predefined data formats
Transaction Enforcement and Financial domain	Smart Contracts
	Native cryptocurrency or a way to simulate currencies or account balance

plications, and cryptocurrencies are a part of this. However, for financial applications to be possible in the supply chain using blockchain, a native blockchain cryptocurrency is not necessarily needed, as cryptocurrencies can be simulated using balances, depending on the design of the blockchain network. For this, and other functionalities to work, smart contract functionality should exist, though. **Thus, the selected framework should either have a native cryptocurrency, or allow for the design of some kind of digital balance or token. Additionally, it should support smart contracts**.

This information is summarized in Table 6.1.

6.1.2 Framework Choice

So, what conclusions can be taken from these framework requirements? Firstly, **a private blockchain framework seems more adequate**, because of all the security control mechanisms that are needed. Secondly, **the framework should allow for highly customizable networks, including not only asset management, but also identity management**, where the participants of the blockchain can be given specific permissions. Lastly, the framework needs to **allow the use of APIs**.

Crossing these requirements with the capabilities of the frameworks presented in the background chapters, we can do an analysis of the applicability of each one:

- Ethereum - It is a public framework that features a native cryptocurrency and has strong financial capabilities. However, it costs money to run code on. Having a lot of the required functionalities on Ethereum would be a lot more expensive than in a private network, where only a few nodes need to be maintained. At the same time, it does not natively feature the essential identity management,

authentication and authorization mechanisms. It features strong financial and traceability, but lacks in security and cost scaling.

- Corda - It is a private network, cheaper to maintain, and most of the privacy requirements are possible. It also focuses a lot on financial transactions, which was a highly rated requirement. However, it sorely lacks in the management of data like assets and entities, therefore having weak traceability properties.

- Hyperledger Fabric - It has the needed mechanisms for authentication and authorization, and, similarly to the other frameworks, has smart contract capability. It is highly customizable, allowing for all the data and identity management needed. As for synchronization, it features easy to deploy rest servers, which is also important. However, it has a setback: though it is customizable, it does not feature a native cryptocurrency, so financial transactions are possible, but only if designed from scratch to be simulated by the network.

From a requirements perspective, Hyperledger fits practically all of the requirements, if we assume that financial transactions can be simulated within the network. Ethereum has the setbacks of cost and security, while Corda has the setback of lacking asset management and traceability.

To make the framework decision final, a performance comparison should also be used. Based on previously published studies, which were mentioned in 2.4.5, Hyperledger has a lower latency, execution time and higher throughput than Ethereum. All while being cheaper to maintain and satisfying more requirements than any of the other analysed frameworks.

The question **"What is the Blockchain tool or framework most adequate to the development of an architecture that can support these requirements?"** can finally be answered. From the analyzed tools, Hyperledger Fabric, together with Hyperledger Composer, seems to be the tool that both satisfies the most requirements and has higher performance and lower costs, being the most adequate of the analyzed tools for the development of an architecture for supply chain.

6.2 Requirements Specification

From the survey, a list of high level requirements was defined, but these requirements need to be refined into more detailed requirements, if they are to give origin to a design and implementation. As it was already mentioned, in theory, the survey would

have to be finished before the requirements were written, but in practice, the work was done more iteratively.

The structure for this specification will somewhat resemble the organization of IEEE 830-1998 requirements specification standard [42], but in a simplified way, without some of the unneeded clutter and information. The specification is divided in the following way:

1. Introduction - Product purpose, scope, overview and users;
2. Requirements - Specific requirements including functional requirements, non-functional or quality requirements, permission list and design constraints.

6.2.1 Introduction - Project Drivers

1. Scope of the Work

This project is being developed as a proof of concept to validate the feasibility of implementing of a supply chain-based blockchain using Hyperledger Fabric and Composer. It does not represent what a fully developed project would act or look like.

Therefore, the objective is to showcase a product concept that can enhance the discipline of supply chain management, according to the results of the survey, by making the data more easily accessible, reducing synchronization time, improving integration and security, and providing a tool that assists in guaranteeing end-to-end traceability.

2. Scope of the Product

This PoC encompasses the development of a Hyperledger Composer business network accompanied by the respective Hyperledger Fabric node topology. The business network itself is comprised of the blockchain ledger model and respective integration endpoints. The ledger will be designed to accommodate transactional data from a supply chain, and it will also be possible to execute smart contracts, in the form of transactions, to manage both assets and the identities of the participants.

The process of extracting data from the ERP systems and forwarding it to the system's node endpoints, as well as the process of sending data from the ledger to the ERP are out of the scope of this PoC. External systems must themselves build their own integration modules. Building the needed APIs for this is a task of this project, as

well as the standardization of the data, with the aim of facilitating future integration tasks.

3. Client, Customer, Stakeholders

This project would benefit any industry which uses supply chain, but more particularly, it would be directed towards any company which wishes to integrate their own information systems (i.e. ERPs and so on) with this ledger, such as to maintain a common underlying information transmission channel with their partners, as well as to have more permanent records of their transactions.

Possible stakeholders for this system and the benefits they have are listed as:

- Supply Chain Executives, Managers and common employees
- Manufacturing companies, Suppliers, Distributors, Retailers - the executives and managers represent all of these company types; these companies can have their partners information more readily available, therefore speeding up the transmission of goods and increasing trust;
- The consumer - the consumers may be able to track some of their goods to a more precise level;
- Auditors and certification Authorities - easy single entry points for auditors to collect their information from;

These are some examples of typical users of the Product:

- Supply Chain members - the employees from all types of companies will be the most common users, and they may register incoming and outgoing products on the blockchain (through their companies own integration module); distribution delivery employees may, for instance, register deliveries, etc.
- Regulatory Entity - the auditors that will use the network to view information from the companies and make sure everything appears to be running correctly and without fraud.
- System administrators - to control the network, help solve any issues that come up and do the needed maintenance.
- Integration developers - the developers in charge of connecting their companies system to this system.

Each user shall have defined roles assigned within the system, according to their needs. These roles, along with the system itself, are the actors of the system:

- System - Hyperledger Fabric and Composer have certain behaviours that can be adjusted, which makes it possible for the system itself to be an actor with specific requirements it shall be able to satisfy.
- Regulatory Entity
- Admin
- Supply Chain Member
  - Supplier
  - Manufacturer
  - Distributor
  - Retailer
  - Customer

6.2.2 Functional Requirements Drivers

1. Functional Requirements

To simplify reading and to save space, the functional requirements list consists of a small explanation at the beginning, followed by each requirement, with the ID and description. Since Hyperledger Fabric and Composer are being used, the requirements have some terms specific to the software, such as asset registry, transaction registry, participants, etc.

The System The system itself needs to be able to record aggregate all of the data into blocks; the data consists of transactions, which are the actions undertaken in the system, and there should be registries for the network participants and assets as well. It is important that the information from these transactions can be made visible and that they can be invoked by the participants which have the permissions to do so. At the same time, the system should be able to handle multiple organizations joining the ledger, inserting all of their current data to the system, as well as extracting it to their own systems when needed.

Identity management, consensus, access control, synchronization of information are all be concerns of the system and how it is programmatically written. Some of

these concerns are already partially supported by Hyperledger, but the remaining have to be implemented. In the end, the system used itself already has a lot of support for the elicited requirements from the survey.

- S1 - The system shall allow the execution of chaincode transactions.
- S2 - The system shall record all user actions, including transactions, into a registry with the identification of the user and action performed.
- S3 - The system shall maintain an immutable list of all the past transactions, in the form of blocks.
- S4 - The system shall allow for the submission of a transaction batch, with many transactions at once (possibly corresponding to the synchronization process of a company uploading their data, for instance).
- S5 - The system shall assign each transaction a timestamp.
- S6 - The system shall allow for the definition of multiple channels/sub-ledgers, to support separate sets of permissions and separate the information of different organizations in Composer.
- S7 - The system shall be able to emit notification events.
- S8 - The system shall be programmed to be able to detect any mismatches that might constitute fraud: for instance, a product being received in a location different than the one where it was supposed to show up.
- S9 - The system shall define different permissions for each role and for some specific identities, including permissions for invoking transactions and reading the data on the ledger.
- S10 - The system shall be able to expose all of the actions that users can undertake in the form of a REST API.
- S11 - The system shall be able to authenticate users through a REST API.

Supply Chain Member For the supply chain member users, all of the following requirements are only doable in case the specific actor has the specific permissions, as was mentioned in requirement **S9**. In this system, the supply chain members need to be able to manage their assets and shipments. All of the actions need to be recorded onto the immutable ledger, which means that any action happens by

the invocation of a transactions. Therefore, the users can invoke transactions for pretty much every action they need to do: to create, edit, delete assets, interact with shipments, deploy contractual agreements, check the status of assets and shipments. All of these actions will ensure both the data management and information tracking and traceability aspects that were so popular in the survey. The financial aspect and enforceable contracts requirements are also included in these requirements.

- SCM1 - The members shall have the ability to invoke transactions.
- SCM2 - The members shall have the ability to read the information of assets, other people and transactions, according to their permissions.
- SCM3 - The members shall have the ability to write and deploy their own contractual agreements on the blockchain.
- SCM4 - The members shall be able to query and obtain the steps through which a particular product has gone, effectively tracing the product from origin up to where it is at the moment of the query.
- SCM5 - The members shall be able to query what is the current entity possessing an asset.
- SCM6 - The members shall be able to create assets.
- SCM7 - The members shall be able to edit or delete the assets they own.
- SCM8 - The members shall be able to create shipments with the assets they own.
- SCM9 - The members shall be able to attach contracts to their shipments.
- SCM10 - The members shall be able to query all the information of a specific shipment, including the owner, holder and assets involved in the shipment.
- SCM11 - The members shall be able to query the shipments owned by a specific user.
- SCM12 - The members shall be able to query the user that possesses a specific asset, if they have the permission to do so.
- SCM13 - The members can check a shipment status, location and the status of all the item status included in the shipment, given that they are either the buyer, seller or current holder of the shipment.

- SCM14 - The members shall be able to submit item damage reports into the system for the assets in the shipments that they hold;
- SCM15 - The members shall be able to update the shipments, along with the status, for the shipments they hold.
- SCM16 - The members shall be able to edit their identity;
- SCM17 - The members shall be able to input an XML file with a standardized format in order to submit data automatically;
- SCM18 - The members shall be able to hold a cryptocurrency, in the form of a balance, with a static equivalence to real money, so that they can transfer it in contracts, in order to enable payments, fines and to settle contractual agreements.

Regulatory Entity

The auditor is a role essential to manual fraud detection, as well as to ensure that the system is working well and as intended. As such, the auditor needs to be able to have full read access to the system.

- RE1 - The regulatory entity shall be able to query and obtain the steps through which a particular product has gone, effectively tracing the product from origin up to where it is at the moment of the query.
- RE2 - The regulatory entity shall be able to query all the transactions, participants and assets in the system.

Admin

The admin is an essential part of the ecosystem that the blockchain will support. Any node maintenance, any problem, will have to be solved by an admin. The admin is also the entry point for users to join the network, as well as an authority that can help fight fraud and make the system more secure and controlled. The identity management mechanisms, which include authentication and authorization are managed by the admin, making the admin an essential user for the enforcement of the elicited requirements.

- A1 - The admin shall be able to revert the effects of certain transactions, by submitting a transaction that has the opposite effect of a given transaction.
- A2 - The admin shall be able to create and delete new ledger channels.

- A3 - The admin shall be able to create a network identity card for a user.
- A4 - The admin shall be able to assign a user's network identity card to an instance of a virtual participant of the network. E.g.John's card ca be associated to the network participant Auditor#21.
- A5 - The admin shall be able to update the details of the participants, including their initial balance.
- A6 - The admin shall be able to create, edit or delete assets;
- A7 - The admin shall be able to submit any transaction;
- A8 - The admin shall have the permission to change the roles of the other users.
- A9 - The admin shall have the permission to give others the permission to change roles.

These requirements were written having in mind the architecture of the Hyperledger projects, as well as the needs of a supply chain. They are also loosely written because they are iterated on during the implementation phase of the proof of concept.

2. Data Requirements

Another important part of the requirements which is not in itself an action undertaken by the users, is the form in which the data is formatted. One of the main objectives of building a blockchain system transversal to a whole supply chain, is to make the data as standardized as possible, so that any organization can easily import or export the data from the ledger to their systems and still have it in formats that any organization's system can understand.

As such, research was undertaken, and the following data requirements were built, based on the standards from the GS1 organization [1], an organization which builds global supply chain standards.

- The data is divided into Master Data and Transactional Data.
- The Master Data relates to Products/Assets and Entities, being more generalist and fixed, for those pairs. It includes the location number (for the location), the

[1] https://wiki.hyperledger.org/_media/groups/requirements/hyperledger_-_supply_chain_traceability-_anti_counterfeiting.pdf

entities name, country and address data, as well as the asset's weight, name, identification number (GTIN).

- The Transactional Data relates to the movement of assets through the supply chain, in the form of transactions. It includes the "when" part of the transactions, such as the planned date, expected dates and actual dates of a shipment, for instance.

In a real system, all these data requirements would have to be implemented in a perfect way, with all the parameters, but this project, only being a proof of concept will use the minimum number of data fields deemed necessary for the implementation and basic functionalities of the system.

6.2.3 Non-Functional Requirements Drivers

1. Non Functional Requirements Drivers

Overall, the product should follow these parameters:

1. Usability

   The available product, corresponding APIs and documentation should be clear enough to allow for the developers to perform the implementation of an oracle, which is a piece of software that connects the blockchain to another external product, serving as a means of pulling and pushing information from and to the blockchain and from and to the external system (ERP, for instance).

2. Performance

   Speed and latency: The throughput and latency on Hyperledger have already been tested, and the throughput is not expected to be as high as in a centralized data system. But, overall, the time to synchronize the information from one company to another might increase; The goal is to make the product be as fast as needed to support the businesses, even if it does not have better performance than other alternatives, since what we are looking for here is the addition of new functionalities (shared ledger);

   Precision and accuracy: The product shall record the data just as it was entered, and the predictions as to whether a product has any mismatching entries shall always be justifiable;

Reliability and availability: The product shall not always be available unless all of the nodes fail at once, which is almost impossible, unless a coordinated attack were to happen; If some of the nodes happen to fail, the response time of the system might be lower than expected;

Scalability: The product should scale to hundreds of companies, which would require a similar number of nodes;

3. Maintainability and portability

   The product is expected to run on Linux based systems, compatible with the Docker, nodejs and golang versions that Hyperledger Fabric uses. More specifically, Amazon Web Services services have servers with the required setup for this.

   Creating new nodes or moving an existing one should be an easy process, without much complication, other than starting the node software on the environment, and closing an existing one, if needed.

4. Security

   Privacy: The system must ensure appropriate visibility of transactions and products, which might be privacy sensitive; sharing some data would pose a threat or could possibly have negative effects for some of the companies; otherwise, transactions should also be secure, authenticated and verifiable;

   Immutability: No one can make changes to the contents of the ledger;

   Authorization: All changes to any data should be approved by the people that possess the data or will be affected by these changes directly. A shipment delivery transaction should, for instance, be approved by both the person delivering and the person receiving the shipment

5. Legal

   The ledger might be subject to verification from competent legal authorities or auditors, and it must also comply with certain laws, such as the european global data protection regulations (GDPR);

   "Traceability and its relation to the law (e.g., regulatory law, international law etc.) has a profound effect upon many products. In the case of diamonds, conflict minerals or rare earth derived material several international agreements and global law governing these items. Ensuing traceability of these products can

literally mean the difference between supporting terrorist groups or supporting those who need good jobs to provide for their family. "

2. Mandated Constraints

- The data will follow the GS1 EPCIS standards [2] and specifications for data formatting, as possible.
- The product shall have well defined APIs that allow for incoming and outgoing data to circulate between the ledger and external systems, namely ERPs.
- Any software frameworks used will be open source. This project will use Hyperledger Fabric and Hyperledger Composer.
- A company/trading partner wishing to join the supply chain's blockchain ledger must comply with the GS1 standards, namely by having a globally recognized identification (GS1 Company Prefix) and the company's locations must also be globally and uniquely identified.

3. Project Issues

1. Open Issues

   The architecture of Hyperledger Fabric and Composer are complex and not completely studied yet, which may lead to deviations from the requirements or alterations at some point.

   Again, these softwares are open source, constantly evolving, and Composer is still in the Incubation phase, so it is not a complete product that can be relied upon for everything. In this case, it is a PoC, so a risk is being taken.

2. New Problems

   User Problems:

3. For a new company that is trying to use the product, they have to take the time to understand the API and develop their own oracle to synchronize all the data.

   Limitations in the Implementation Environment:

[2] https://www.gs1.org/sites/default/files/docs/epc/EPCIS-Standard-1.2-r-2016-09-29.pdf

4. It is not yet known how many companies this will be able to scale to, since it also depends on how many nodes each of them run, and the quantity of data going through the network at any given moment.

5. Costs

   Maintaining the blockchain running requires little to no cost, as only a few dozen nodes might be needed at most, depending on the scale, and each of them might be operated by different companies.

   The development itself and testing might run into cost troubles in the aspects that require the simulation to be faithful to reality, such has having lots of servers distributed geographically running at the same time (requiring expensive cloud services, at times).

6.3 Design and Implementation

Although some projects build entire or partially custom blockchain network software, using a framework like Hyperledger Composer is a much more streamlined way to get a fast functioning prototype.

The baseline for the design will be the written list of requirements, mainly the functional requirements part, and the specifications of the Hyperledger projects.

The design can be divided into 4 parts:

- Model Design for a Composer Business Network
- Access Control Design and Identity Management
- Network Topology and Deployment
- Integrating Existing Systems and Building External Applications

These parts are not necessarily sequential, but following the listed order may bring about the best results in development. Since time was an issue and the dissertation already included aspects other than the development, the scope of the development itself could not be broad enough to include a deep exploration of all the aspects listed.

Therefore, the project here presented focuses more on the quality and functional aspects of applying blockchain to the supply chain, and not so much on the quantitative part, which would include tests to the efficiency of the network (throughput, latency). What this means is that the development had a bigger focus on the model design,

access control design,identity management and the implementation and validation of these aspects than on building and testing a realistic node topology. The scope for the last item, integrating existing systems and building external applications,was also narrowed down to the essential.

This section is divided in several parts to explain in which way these items were approached, so that conclusions can then be taken to reach an answer to the question: **"Is it possible to build a feasible architectural design, by using such a tool, to implement all these requirements?"**

6.3.1 Composer Business Network - Model Design

From the requirements, the business network model for Hyperledger Composer was designed and built. This includes the specifications for the *.cto* file, with the definitions of all class types participants, assets, transactions and events, as well as the enums and concepts (which are basically non-instantiable data types). The script file with the implementation of the transactional code and a query file with custom queries for the blockchain are also explained.

Only the most important data attributes for the classes will be explained, even though **all of the data attributes from all the classes can be seen on the class diagram exhibited in the annexes.**

Participants

The first step towards designing the system was to define who the users are, in order to model the participants. This is an easy task, as the actors of the system were already defined beforehand.

The logical separation for the participant type:

- auditor - the participant type that will represent the auditor role; the reason a specific participant type is needed for this actor is so that the access control for the auditors can later be specified;
- supplyChainMember - the main participant of the network, the supply chain members are the actual users that will be interacting with each other, so it makes sense that they are modeled; Sub-types were also designed, having in mind that they may exhibit different behaviours, and different access rules can be written based on the sub-type;

- Supplier
- Manufacturer
- Distributor
- Retailer
- Customer

Both the auditor and the supply chain member participant types have as attributes some company and personal identification, and the supply chain members also have an account balance, which is the basis for financial transactions. The reason the admin does not appear here is that the admin of a business network does not need a user type to be able to invoke transactions. Instead, they only need their admin identity card, which in no way needs to be tied to a network participant.

Assets

Another integral component of the system are the assets, for the asset management aspect of the supply chain. If traceability of products is to be achieved, these products need to be modeled as assets, so that the network can register which ones exist, their status, and any changes that happen.

The proposed assets in this model were: **Commodity**, **ShipmentBatch** and **Contract**. Each of them serves a purpose. A diagram with the assets and their relationships to the participants and each other can be found in Figure 6.1.

Commodity - Represents a single product being exchanged, with all its attributes, including the product ID (GTIN), name, description, item status and even the ID of the person who owns it (a participant of the network).

ShipmentBatch - Represents a physical shipment, that a buyer orders from a seller. The shipment includes all the tracking information, including tracking number,a shipment status and location. Additionally, the shipment has an array of the Commodity items included in it, as well as information on who is the current physical holder of the shipment and owner (other participants). When a shipment is created, since it has an origin and a destination along with other sensible data and shipment conditions, it makes sense that a contract is generated, therefore the shipment also has a contract associated.

OrderContract - This represents a digital contract for the conditions of the shipment. The expected arrival location and time, the buyer and seller, as well as a payment price, for when the shipment is delivered (though it is optional). One of the requirements was

that contractual agreements should be made possible, as well as financial transactions, and this is the proposed way to make it happen. Fraud checks can also be done on the arrival location, to check if the real arrival location matches the expected one.

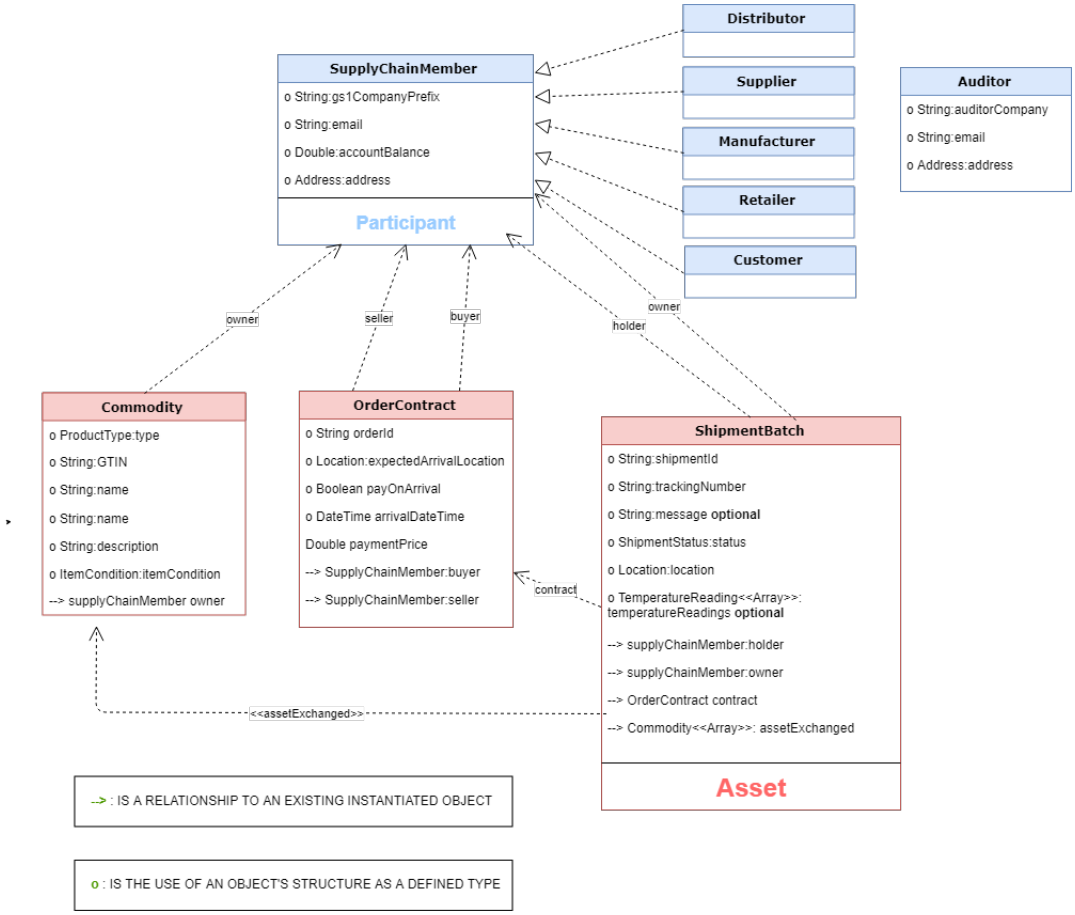


Figure 6.1: Class Diagram for the relationships between assets and participants.

Transactions and Transactional Scripts

The way that the participants an interact with the network and with the assets is through transactions. **Transactions are the chaincode functions with certain parameters invoked by a user of the network** (how a user can invoke a transaction is another matter, also discussed in the following sections). So, while the participants and assets model the users and data storage of the system, **the transactions model the behaviour of the network**, by accessing the participant and asset registries and making changes. Any transaction invocation is always recorded in an immutable historical record.

All the creation, deletion and update transactions are already supported by default by Composer, but all other behaviour needs to be modeled, and sometimes, even these

default transactions should be made restricted and replaced by custom made ones, to ensure system integrity.

The transactions that were modeled for this network, are featured in the following list, with their descriptions:

- **CreateShipmentAndContract** - creates a shipment and the associated contract at the same time, for a certain set of parameters, which includes a buyer, seller, the commodities to include in the shipment, etc.
- **UpdateShipment** - update a shipment's tracking status, location and optionally pass it over to a new holder. In case the shipment is being delivered to the buyer, if a payment was established in the contract, it is automatically done. **Emits a "possible fraud" event when a shipment does not arrive on the expected location, and a normal event warning that the shipment was updated as well**.
- **UpdateCommodity** - A custom transaction to update a commodity, instead of using the default one. The reason is to make access control rules easier for this behaviour easier.
- **DeleteCommodity** - Again, a custom transaction, to delete a commodity. The reason for this transaction being custom is that a verification must be done to check if the commodity is part of a shipment before deleting it. In case it is, to maintain the integrity of the system, the commodity is not deleted
- **ReportDamagedGood** - Sometimes, goods are damaged during their shipment. This transaction updates the current status of a commodity and a description of the occurrence, so that a registry for any damages occurred can be
- **TransformCommodities** - This transaction converts 1 or more commodities in a different set of commodities, by deleting the old ones and creating new ones. It is useful, for instance, when creating a product out of raw materials. **Emits an event warning that the products were transformed**.
- **TransferCommodityPossession** - transfer the ownership of a commodity to a different participant of the network, but only if the commodity is not currently part of any shipment. **Emits an event warning about the change of ownership**.
- **TemperatureReading** - Finally, and to ensure full product condition traceability, temperature readings can be inserted into the system. A shipment can log an

array of temperature readings, and each time this transaction is called for a shipment, a reading can be inserted into the array.

Additionally, the transactions, with all their parameters can be seen in Figure B.1, from appendix B.

Queries

The transactions are used to model actions and behaviour, but to retrieve some of the information, something else is needed. **Composer already retrieves the basic information for participants, assets and transactions, but anything else requires queries**. Hyperledger Fabric uses a relational database to store the asset and participant registries and Composer features a way to retrieve that data easily, through filtered queries (using a javascript framework, LoopBack).

In this proof of concept, some queries were programmed for some of the most important pieces of information that can ensure traceability and other properties. These can be found in Table 6.2.

Table 6.2: List of designed and implemented queries, with their parameters.

ID	Name	Retrieve	Parameters
1	selectShipmentByID	ShipmentBatch	Shipment ID
2	selectShipmentByOwner	ShipmentBatch[]	Shipment Owner
3	selectShipmentByHolder	ShipmentBatch[]	Shipment Holder
4	selectShipmentByCountry	ShipmentBatch[]	Location Country
5	selectShipmentByTrackingcode	ShipmentBatch	Tracking Number
6	getHistorianRecords	Transaction[]	None
7	getHistorianByPerson	Transaction[]	Participant
8	getHistorianByType	Transaction[]	Transaction Type
9	getDamagedGoodsTransaction	Transaction[]	None
10	getCreatedShipmentsTransactions	Transaction[]	None
11	getCommoditiesTransformations	Transaction[]	None
12	getCommodityOwner	Commodity	GTIN
13	getShipmentWhereCommodityExists	ShipmentBatch	Commodity

The queries are pretty straight forward to code. They are a basic SELECT type of query, with the specification of what object we want to select, followed by a WHERE kind of statement, where certain conditions like equality, inequality, *"CONTAINS"* among a few others can be used.

For instance, query number 12, *getCommodityOwner*, takes as parameter the Commodity's ID (GTIN), but instead of retrieving the owner of the commodity, a participant,

it returns the Commodity itself. Why? Because the commodity retrieved has all the information of who the owner is. This information has to be extracted manually since it is **impossible** to make a query to select only that attribute.

Query Difficulties

Though the ease of coding the queries is appreciated, they are pretty limited in what they can do. For instance, it is impossible to natively nest a *"SELECT"* inside another. Since the database behind is not relational, there are also no powerful query elements like *"JOIN"*, which makes for getting some pieces of information very hard or even impossible.

To make up for this, some changes in the model had to be made. Originally, the *Commodity* asset did not have a *Owner* data attribute, since it was not thought to be needed. A *ShipmentBatch* has a *Owner*, and a list of *Commodities*. Therefore, if a relational database were used, by providing the ID of a *Commodity* as parameter, it would be possible to retrieve the *Owner*, simply by joining the tables for the shipment, participant and commodity using the given ID, and it would return the owner. In a relational database, any relation points both ways, so it is easy to navigate between the classes of a model, but that does not happen here.

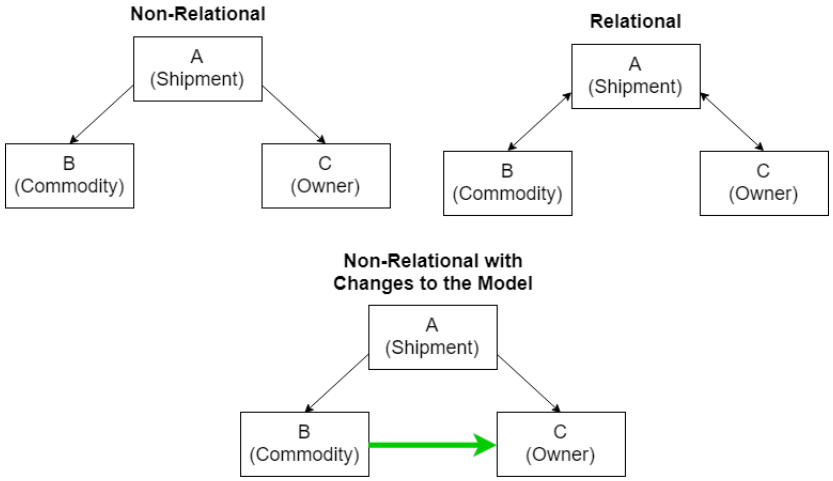


Figure 6.2: Exemplification of the non-relational model problem in the Hyperledger Projects.

Figure 6.2 illustrates the problem anc changes to the developed model that had to be made to adapt to it. The *Commodity* was adapted to have a *Owner* data attribute so that it is easier to query the owner of a commodity. This pretty simple query would be a lot harder to accomplish in Composer, and anything more complex or having more tables than this would be pretty much impossible to do without exporting all the data to another system that could organize it and query it in a different way.

Another problem that this can bring, other than having the trouble of changing the model, is the integrity issues that can be caused in the data. Anytime that the owner of a shipment is changed, the owner of the commodity also has to be changed in the code. If the owner of a commodity was not changed in the code, then it would be inconsistent with the owner of the shipment, who in reality should also own the commodity. **If a programmer consistently adds attributes to classes to make up for the weak Composer queries, eventually they might forget to make a change similar to this one and the system might lose integrity**.

6.3.2 Composer Business Network - Identity Management and Access Control

Creating a model design for the network with all the previously explained items grants the needed basic functionalities for the system to work, but it misses one of the most important aspects from the requirements: **the security and access control**.

As previously explained in Section 2.4.3, Composer features the concept of Participant, a network instance of the class that represents a user, but it also features identity cards, which are files that function as the private key and identification card for a real person. An identity card and an instance of a participant can be linked to each other.

The control rules are written to reflect what actions are allowed to certain participant class types, to specific participant instances, and even to specific identity cards.

A set of access control rules was designed with the requirements in mind, so that only certain people can access certain sensible pieces of information, and so that not everyone can manage other people's data.

All of these rules were implemented in a mixture of two methods. The first is by using the access control composer language and *.acl* file. The second is to program the transactional scripts to check the identity and data of the person invoking the transaction and then throw errors if the invoker should not have access.

The designed permissions are sectioned by items, namely the **transactional data and invocations**, the **asset data and its CRUD actions** and the **participant data and its CRUD actions**. The following list features a series of these items, with their name, action and access control rule. The admin identity card is has full access, so it will be mostly omitted from these rules.

Permissions and access control rules:

Transactions - Invoke

- CreateShipmentAndContract - Allowed for every participant and identity, excluding the **customer** participant types;
- ReportDamagedGood - Allowed only for the holder of a shipment with the specified commodity;
- TemperatureReading - Allowed only for the holder of the shipment with the temperature being read;
- TransferCommodityPossession - Allowed only for the owner of the commodity.
- TransformCommodities - Allowed only for the owner of all the commodities that are input arguments.
- UpdateShipment - Allowed only for the holder of a shipment;
- UpdateCommodity - Allowed only for the owner of a commodity;
- DeleteCommodity - Allowed only for the owner of a commodity;

Assets

- Commodity
  - Create - Allowed for supply chain member participant type;
  - Read - Allowed for supply chain member participant type who owns the commodity being read; allowed for the auditor participant type;
  - Update - Allowed for supply chain member participant type who owns the commodity being Updated;
  - Delete - Admin only;
- OrderContract
  - Create, Update, Delete - Admin only;
  - Read - Allowed for the buyer and seller of the contract; allowed for the auditor participant type;
- ShipmentBatch
  - Create, Update, Delete - Admin only;
  - Read - Allowed for the owner and holder of a shipment, as well as the buyer in the shipment's contract; allowed for the auditor participant type;

Participants

- Supply Members
  - Create, Update, Delete - Admin only;
  - Read - Allowed for the auditor participant type; allowed for a participant's own details;
- Auditor
  - Create, Update, Delete - Admin only;
  - Read - Auditor can read own details;

So, these rules reflect some basic facts. Auditors can read anything in the network, while admins can read, update, delete, create or invoke anything. The common users are subject to reading only details from themselves and the assets they are associated to in some way (buyer, owner, holder, etc). In the same way, they are also limited on the transactions they can invoke based on these associations.

Finally, the CRUD actions that the supply chain members can execute for the assets and participants are also pretty limited, in order to **maintain system integrity**. This was essential since Composer does not, by default, make some essential integrity checks on the CRUD actions. For instance, when updating an asset's reference to another asset, Composer does not verify if the new reference actually points to an existing instance of an object, which is the reason for why some custom transactions were designed, such as *UpdateCommodity*.

Though all of these access rules were designed, not all were implemented in time, but the ones that were implemented were tested and worked flawlessly.

Conclusions and Security Concerns

In an ending note, identity management is a complex thing in the Composer framework, not only because of the rule design, but also because of how it can be bypassed by **human error and social engineering**. Nothing, but the immutability of the ledger records, is granted, especially when there is an admin that can control the system.

Additionally, some transaction invocations would be more secure if they could have a multiple signature from many parties. One such transaction could be the *UpdateShipment* transaction. when an item is being delivered to its buyer, it should be made sure that the transaction was acknowledged by both parties, so that the payment

could be confirmed securely. **Such a thing is not yet permitted by this framework and constitutes a grave security fault, therefore not satisfying the security requirements completely**.

6.3.3 Network Topology and Deployment

Composer also features network topology design, which includes choosing the organizations running the network, how many nodes each organization hosts and what types of nodes and the endorsement policy. This is especially useful to quantitatively test the efficiency metrics and scaling of the system, depending on the topological setup. Such measurements are important to compare this system with other possible architectures.

However, the scope of this project covers mostly the qualitative and functional part of building the network, with the aim of finding out if the requirements can successfully be implemented and work as intended, independently of how the network scales and independently of how other architectures behave. Especially in the case of Hyperledger Composer, the implementation so far works, either with just 1 node,nd, even though there was no implementation of a big topological network of nodes, some speculation or with a small number of them.

With that in mi and thought were given as to how a proper network should look like. Figure 6.3, taken from the Fabric documentation, exemplifies how a small supply chain topology would be organized.

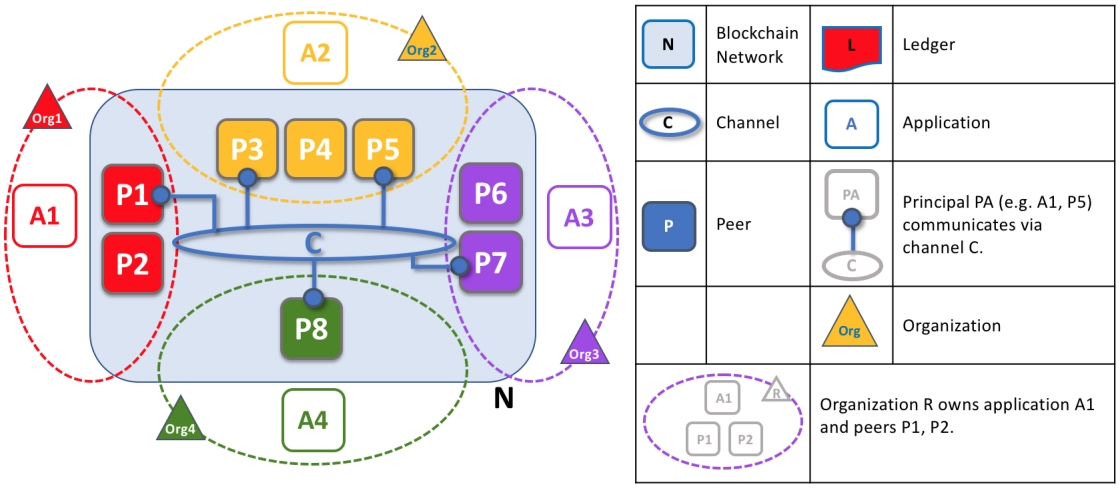


Figure 6.3: Example fabric topology for 3 organizations. Adapted from [43]

6.3.4 Integrating Existing Systems and Building External Applications

The synchronization requirements were another strong focus points, an issue that deals a lot with building points of access for several external applications to use while importing, exporting data and interacting with the blockchain.

Architectural Context Considerations

An important task of architectural design is identifying where blockchain is in the middle of everything. It was already laid out that it can be used by other applications as the source of truth. **Architecturally, this may mean that blockchain might be used as the lowest layer** of a network, and that it can even be the support for more than 1 network.

All the other data layers would get the information from the blockchain, and inject their information to the blockchain, when needed. The upper layers could organize and treat information however they want, as long as they could retrieve it securely. This way, it would even be possible to make up for the shortcomings of the query language of Composer, by retrieving the Blockchain data to a higher level layer application and have that application organize the data in a way that would make it more accessible. Of course, building these upper layers is all up to the developers of each company and application that wants to interact with the blockchain.

Figure 6.4 shows an idealization of what such an architectural layer design would look like. From oracles extracting and inserting big chunks of data periodically, to applications using the blockchain directly as their data storage source, the possibilities for the layers above the blockchain network are many, and not even limited to what can be seen in the figure.

The integration part of this project deals with defining secure and authenticated points of access for the upper layers to have access to.

REST Server API and Authentication

Composer already features a ready-to-start API server, which includes all the transaction invocations, queries and CRUD operations. Authentication, using identity card files, is also included in the REST server's functionalities.

It is also possible to write a custom REST server for composer, using Angular applications, but the default server has the necessary features.

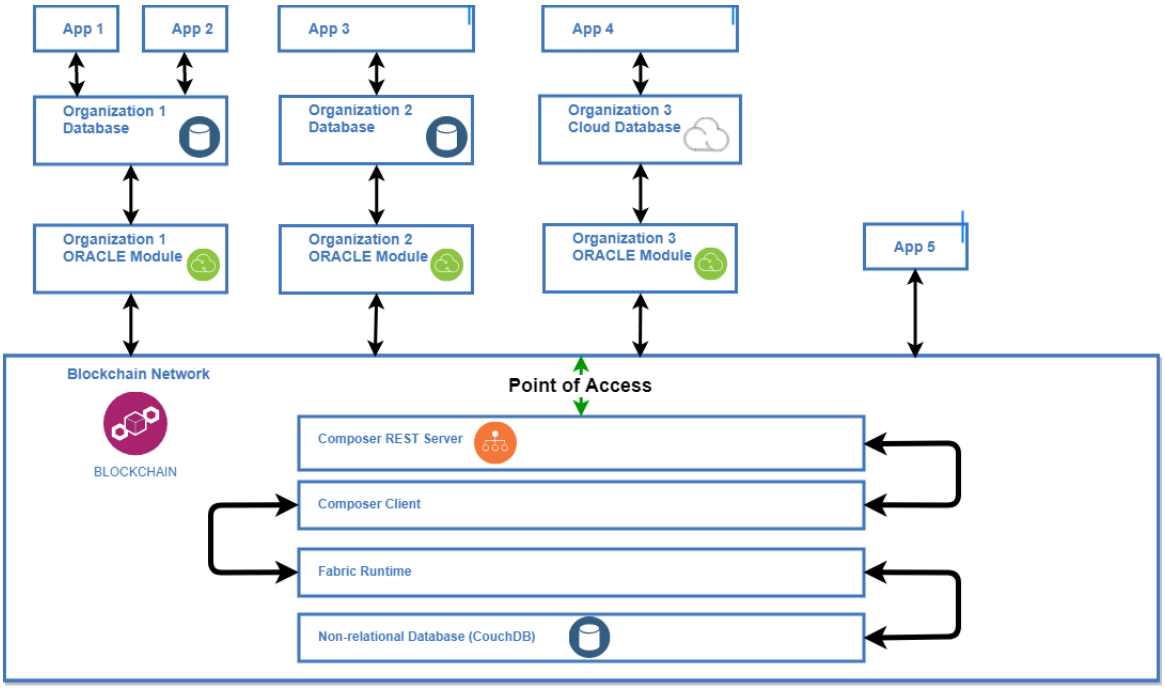


Figure 6.4: Architectural layer representation of Blockchain Integration with other systems.

Other Applications

Other software are possible to be implemented with the blockchain. One such instance of a software is Node-RED, to connect the blockchain to an IoT network of devices, especially important in case we want to achieve traceability of product conditions or quickly scan products. It can also be used to subscribe to network events.

6.4 Results and Validation

6.4.1 Requirements Validation

In order to determine the answer to the third and last of the problem statement questions, *"Is it possible to build a feasible architectural design, by using such a tool, to implement all these requirements?"*, the suggested design and its implementation have to be evaluated in an objective way.

The proposed methodology is to validate both the functional and non-functional requirements from the specification.

As for the functional requirements, it needs to be ascertained which ones were possible to implement, which ones were not, which ones were indeterminate (as in, the development was not finished in time to come to a conclusion as to whether they

were possible) and which ones are only possible partially.

As for the non-functional requirements, a few comments will be made regarding how each point was approached and whether it was fully satisfied, partially satisfied or not satisfied at all.

Functional Requirements Validation

As can be seen in Table 6.3, a few of the functionalities are either not possible, or only in a limited way.

Table 6.3: Validation of the functional requirements, according to the possibility of their implementation.

S1	Allow chaincode transactions	Possible	SCM10	Query specific shipment	Possible
S2	Record all user actions	Possible	SCM11	Query shipment owned by a user	Possible
S3	Maintain immutable list of transactions in blocks	Possible	SCM12	Query an asset's owner	Possible
S4	Submit transaction batches	Not Possible	SCM13	Check shipment status, location, and all of the item's status	Possible
S5	Assign timestamp to transaction	Possible	SCM14	Submit item damage reports	Possible
S6	Have multiple ledger/channels for Composer	Not Possible	SCM15	Update a shipment with a new status and location	Possible
S7	Emit notification events	Possible	SCM16	Edit own identity	Possible
S8	Detect fraud mismatches	Possible	SCM17	Input an XML file to submit data	Partially
S9	Define different permissions	Possible	SCM18	Hold a cryptocurrency or enforced balance with equivalence to real currency	Partially
S10	Expose REST API	Possible	RE1	Query all the steps of a product's life	Partially
S11	Enable REST Authentication	Possible	RE2	Query every system registry	Possible
SCM1	Invoke transactions	Possible	A1	Revert transactions, by submitting an opposite transaction	Not Possible
SCM2	Read assets, participants, transactions, according to permissions	Possible	A2	Create and delete new ledger channels	Partially
SCM3	Write and deploy contractual agreements	Partially	A3	Create network identity cards	Possible
SCM4	Query all the steps of a product's life	Partially	A4	Assign identity cards to participant instances	Possible
SCM5	Query owner of an asset	Possible	A5	Update details of participants, inc/ balance	Possible
SCM6	Create new assets	Possible	A6	Create, edit, delete any asset	Possible
SCM7	Edit and delete owned assets	Possible	A7	Submit any existing type of transaction	Possible
SCM8	Create shipment with owned assets	Possible	A8	Change a user's role	Possible
SCM9	Attach contractual agreement to shipment	Possible	A9	Give other users permission to edit roles	Possible

There were 3 requirements not possible to implement, all due to limitations of the framework: submitting transaction batches, using multiple ledgers on composer and reverting a transaction's effect.

- **S4 - Transaction batches** would have to be implemented on the client side of any application that wants to communicate with the blockchain, and even then, it could only "simulate" this feature, because it would have to submit the transactions one by one, and they might not all get submitted in the same block, which can be problematic.

- **S6 - Multiple channels** are still not available on Composer, making the creation of multiple channels for groups of organizations impossible for this framework.
- **A1 - Reverting a transaction's effect** is not possible, because it is not possible to programmatically in the script file, using the Composer runtime API, to access the contents of a previously submitted transaction. Composer only makes this content accessible to the Client API to be used by other applications.

The partial requirements mean that part of the requirement was possible, or the requirement was somehow simulated in a different way than the one written in the specification.

- **SCM3 - Deploying contractual agreements** is only possible according to the contract format specified in the design. This means that custom contracts can not be submitted. Smart contracts are also limited to the ones already deployed on the network, and only the admins can update the network with new contracts.
- **SCM4 and RE1 - Querying a product's lifecycle steps** is possible for commodities that have not gone through transformations, by simply querying all transactions associated with that commodity. However, when a commodity is transformed, it basically is deleted and new ones are created and there is no way, with the current Composer queries, to retrieve all of the transactions associated with all the products, from before and after the transformation.
- **SCM18 - Holding a cryptocurrency** is simulated here by the balance, reason why it is only partially implemented. It still has the same usability a real cryptocurrency would have, since the balance can not be double spent because of the consensus, but it has limited interchangeability with other currencies and there are little management functions for the balance.
- **A2 - Create and delete new ledger channels** is possible in Fabric, though not in Composer. In theory, an admin can create a channel for Hyperledger Fabric, but it will not be usable by Composer.

Non-Functional Requirements Validation

The validation and analysis of the satisfaction of the quality requirements is presented in Table 6.4.

By joining the conclusions reached in both the functional and non-functional requirements, there is a better understanding towards validating the requirements

Table 6.4: Validation of the non-functional or quality requirements.

Usability		The APIs are pretty straightforward to use. It is divided in assets API calls, participant calls, transaction calls and query calls. It should be possible to develop oracles and software integration modules with relative ease.	Satisfied
Performance	**Speed and Latency**	Though not formally tested, the speed of the system was informally asserted from the software tests performed. Composer seems to consistently slow down the speed of the system and have a higher latency when compared to the standalone use of Hyperledger Fabric (based on the metrics analysed in the background)	Not totally satisfied
	Precision and Accuracy	All data entries are accurate, and there is some degree of fraud detection/user mistake detection.	Satisfied
	Reliability and availability	These factors work as expected. The more nodes, the better the reliability, but, depending on the endorsement policy, some transactions might not go through when specific nodes are down.	Partially satisfied
	Scalability	Technically, it is possible to scale the product to the levels of hundreds of companies, though the performance suffers a degradation.	Satisfied
Maintainability and Portability	**Hardware**	It is easy to add new nodes and edit the configuration.	Satisfied
	Software	Limited maintainability, since the new versions of a business network have to be compatible with the previous ones, and for this, some parts of the design can not be removed.	Partially satisfied
Security	**Privacy**	The access control rules ensure privacy.	Satisfied
	Immutability	The implemented blockchain designed is fully immutable.	Satisfied
	Authorization	Some actions should require permission from more than 1 user, which does not happen.	Partially satisfied
Legal		The new global data protection regulations (GDPR) put the legality of blockchain into question. Supposedly, GDPR argues that any data containing personal information must be removable. This rule is, however, incompatible with the immutability requirement.	Not totally satisfied

from the survey and also reaching a final conclusion for the last questions. The validation of the survey requirements is shown in Table 6.5.

As can be seen, most of the requirements were fulfilled, but not all of them totally.

Security, as could be seen in the functional and non-functional requirements, has some faults.

Regulatory auditing is also limited, because of the commodity transformations, which limit the scope of how far we can trace back a product.

The development of industry standards is done by organizations such as GS1, which are actively working to ensure that the standards they develop are adopted world-wide.

Financial transactions were developed, but not fully to a level that can be used industrially, so there is a lot of work to do on that field.

Enforceable contracts, as explained, are also limited to the designed contracts, and the deployment of new or custom contracts is limited.

6.4.2 Development Limitations

In retrospective, the chosen framework had a big impact in the implementation of the requirements. Some were made easier, but others were harder or even impossible, due

Table 6.5: Validation of the survey elicited requirements.

Security	Security according to the latest requirements	Limited
	Controlled access for the users	Satisfied
	Secure data storage	Satisfied
Traceability	Regulatory auditing	Limited
	Fraud detection	Satisfied
	Asset management	Satisfied
	Real-time tracking information	Satisfied
Synchronization	Interoperability between systems	Satisfied
	Development of industry standards	Limited
	Real-time sharing of information with partners, leading to better working relationships	Non-applicable
Transaction Enforcement and Financial domain	Financial transactions	Limited
	Enforceable contracts (smart contract functionality)	Limited

to limitations in the software, and not due to the design.

Some of the Composer framework limitations found that impacted the development of this proof of concept:

- Queries are not powerful enough. There is no *JOIN* and the usual SQL syntax is not always applicable. There is also no *WHERE ... IN ...*, for instance. Model changes had to be made to adapt to this, and integrity might be lost.
- The default Composer REST server is somewhat limited in its API. To extend the API, a custom server needs to be written; The queries and use of LoopBack filters is also limited on the default server, but can be more extensively used by building a custom server.
- It is impossible to natively nest queries with the query language. To make up for this, several independent queries sometimes need to be executed, which is very inefficient.
- Some queries are bugged on features like *LIMIT*.
- The run-time API does not have methods to access transactions, so that past transactions from the registry can be interacted with in the script file.
- Some access control rules might be impossible to apply to certain queries.
- The framework is still in development and has some bugs. For instance, *LIMIT* in queries is bugged and does not work. Sometimes, when an asset is deleted,

a new asset with the same ID as the previous can not be created, because the system thinks the previous asset still exists.

- Organizations are not supported in Composer in the same way they are in Fabric.
- Composer somewhat lacks integrity checks when references to instances of other objects are used.

6.5 Conclusions

The whole chapter focused on building a list of requirements, designing a system, implementing it and validating the work done. All to answer the question: *"Is it possible to build a feasible architectural design, by using such a tool, to implement all these requirements?"*.

Thus, the final results and validation are essential to the answer and they are the ultimate goals, but the whole process of getting there by building the requirements, the design, and the implementation also provides important contributions.

Through the analysis of the results, the direct answer to the posed questions is: *No*. It is not possible to use the Hyperledger Composer and Fabric frameworks to satisfy **ALL** of the requirements through the built design, mainly because of limitations in the software, and not because of the architectural design itself.

Even though the answer to the posed question was negative based on the results from this dissertation, the fact remains that it was still possible to implement most of the requirements. Therefore, if the question is instead changed to *"Is it possible to build a feasible architectural design, by using such a tool, to implement **MOST** of these requirements?"*, the answer turns into a **YES**.

The design itself is feasible to be implemented, and even though the tool has some limitations, most of the functionalities and quality requirements were indeed satisfied. Both the design and the tool are not without some merit and it would not be fair to simply say that such an architectural design, based on Blockchain, is not feasible or usable simply because it does not satisfy an extensive list of requirements.

Chapter 7

Conclusions

The work done so far can be aggregated to reach some conclusions. This chapter summarizes these conclusions to reach a final conclusion in the overview. The difficulties that molded the trajectory of this dissertation are alo described.

Finally, this chapter features important content in the form of a list of contributions of this dissertation, which interlace with the possible future work. More research can be done in this area and hopefully some of it based on the findings from this thesis.

7.1 Overview

All the work and contributions of this dissertation aim towards proving whether the following statement, introduced in Chapter 4 is valid or not: *"**Blockchain is a good architectural design for the Supply Chain Management domain**"*.

For this reason, 3 questions were formulated in the problem statement:

1. **What supply chain issues, improvements and requirements do the experts really find the most important?**
2. **What is the Blockchain tool or framework most adequate to the development of an architecture that can support these requirements?**
3. **Is it possible to build a feasible architectural design, by using such a tool, to implement all these requirements?**

Chapter 5 provided the answer to the first question. Chapter 6 used this answer to provide the answer to the second and third questions, which can now lead to a final conclusion towards the initial statement.

Thesis Statement

We have almost all of the pieces to reach a final conclusion towards the main thesis statement. However, there is still the need for the expression "a good architectural design for the SCM domain" to be clarified. **What exactly does it mean for an architectural design to be good for some domain?**

If we were to compare a new architectural design against other existing designs for that same domain, then a system based on a new design could be considered good if it could cover all of the functionalities of the other designs and have some additional ones or be more efficient. But it was already determined that **the focus of this thesis statement was not to compare this architecture to others, but to evaluate it according to the requirements**. Therefore, a good design could be one that objectively covers all of the most important elicited requirements.

The chosen framework had the possibility of satisfying the most requirements, out of all the frameworks. Even so, by answering the last question, it was concluded that the proposed design, even while choosing the most convenient tool possible, could only fill the requirements partially.

Therefore, if a good design is one that must be able to satisfy all of the requirements, then it is not certain that Blockchain is a good architecture for the supply chain management domain, since not all of them were satisfied.

However, this does not mean Blockchain is not useful, since it can still partially fill some important requirements. **The fact that by itself, it might not be good enough to satisfy all the requirements, does not mean it cannot be used together with the current architectures** to fill in some gaps. Blockchain can still be applied as an aid and enhancement to some information system functionalities. For instance, financial transactions was a requirement that was fulfilled and listed very high on the requirements from the survey. No other present architecture uses this functionality, but through Blockchain, it can be implemented.

7.2 Contributions

It is important that a conclusion was reached, but it is also important to look at the work was done to reach that conclusion. The whole process of originating the answers to the questions that support the conclusion also generated useful contributions.

These contributions might be useful for future research:

- **Survey Research and Analysis of Expert Opinions**

  - Validation by the experts of the most important supply chain issues, supply chain points of improvement, information system features that a supply chain needs and the most wanted blockchain use cases for supply chain management
  - List of requirements for a blockchain-based supply chain management software, grouped by area

- **Software Engineering Artifacts**

  - Framework Evaluation and Architectural design, including a Hyperledger Composer model[1], a Prototype implementation and a validation of the feasibility of the requirements

7.3 Difficulties

The work done in this research was not always linear, even though the logic is made to be linear. One of the initial difficulties that most impacted the research was the lack of a quantitative baseline to compare the proposed design to. There are many other projects that attempt to, in one way or another, enhance the capabilities of supply chain by applying blockchain as an architecture, some even showcased in the literature review. However, none had publicly available efficiency metrics that could be used for this project to be compared to. Many developers and companies were contacted to try to collect more data from the existing projects, to establish a baseline but the contacts did not fall through in a satisfactory way.

In the same way, some supply chain companies were contacted to the purpose of establishing a quantitative baseline for the needs of supply chain. This could have included, for example, knowing how many products a company shipped per a certain amount of time, since this data could lead to interesting conclusions. But none had the specific data needed or the interest to maintain a continuous working relationship for the purposes of this research.

Thus, one of the biggest difficulties for this dissertation was the lack of partnerships and difficulty in establishing and maintaining contacts interested in contributing to the research.

Another difficulty was that the frameworks used are not very mature. Some had their development abandoned, while others are still in development, with unstable

[1] The code developed for this model is open source, and can be found at https://github.com/coletiv/supplychain-composer-thesis.

code, and there are always new versions being released, which sometimes changed the way the features worked.

Finally, without a comparison baseline, the survey was developed, as a way to get expert knowledge in a more indirect way. But getting experts to answer the survey was not easy. The expected profile for the participants of the survey was not a common one, and the survey had to be shared through specific channels, often times having a relatively low answer rate.

7.4 Future Work

The work here done was introductory in nature, and explored the requirements and their satisfaction as a whole. Additionally, the maturity of the tools used was questionable. This research can be used as the groundwork for more important conclusions and contributions to be reached.

Future work might include:

- Gather requirements through other means, such as interviews and top-of-the-market tools for supply chain management.
- Taking specific requirements from the most important requirements on the list and research on using blockchain to successfully apply them as enhancements:
  - Financial applications, payments and contractual agreements.
  - "Applying Blockchain as a financial alternative in the Supply Chain", for instance, can be a topic of research, among others.
- Research and attempt to use different frameworks to fulfill the same requirements that were elicited in this dissertation.
- Build a more complete topological network for the proof of concept, thoroughly analyze the performance and compare it to other systems and architectures, including systems different from Blockchain.

Appendices

Appendix A

Statistical Analysis Methodology

Table A.1: Data analysis metrics used in the survey results analysis

	Metrics	Meaning
Measures of Central Tendency or Location	**Mean**	The mean represents the most probable value. In this survey, with the use of scales with a lower and upper bounds, the mean has the roles of representing the average value of agreement, importance and other measures.Normally, when the skewness of a distribution is high, the meaning of the mean may get distorted by the existence of outlier values. However, since the scales have a low range, with a lower and upper bound on the answer values, this is not much of a concern. Therefore, even in cases of skewness, the mean can be a useful metric.
	Median	Value of the 50% percentile, for numerical answers. Half the answers are above this value and half are below, pointing to a central tendency around this value. This is a good metric to use, especially in skewed distributions where there are outliers in the collected values, since the meaning of the mean may get slightly distorted by the outlier values.
	Mode	Most frequent response. Though it represents the most popular answers, by itself the metric means nothing else, as there might be answers almost as popular or not.
Measures of Spread, Scale or Dispersion	**Standard Deviation**	Quantifies the variation within the data set, by showing how much the distribution spreads to either sides of the center. A high value for the standard deviation means that there are a lot of values away from the mean, from which can be concluded that there is not a general consensus on a certain answers.A low value means that there is consensus, since all the values of the data set are bundled more closely together.
	Range	Difference between highest and lowest value of the data set. Together with the standard deviation, indicates the dispersion of the value of the answers. A range of 0 means that a question had the same value for all responses, for instance. This metric ignores the frequency with which each answer was given, that is why it must be coupled with the standard deviation to be relevant.
Measures of Skewness and Kurtosis	**Skew**	This metric indicates the lack of symmetry in a distribution, where the results bunch up in one side of the distribution. For instance, negative skewness values indicate a skew to the left: the values bunch up at the right end of the distribution and the left tail is long, indicating there are outliers in the lower values.

Appendix B

Composer Model Class Diagram

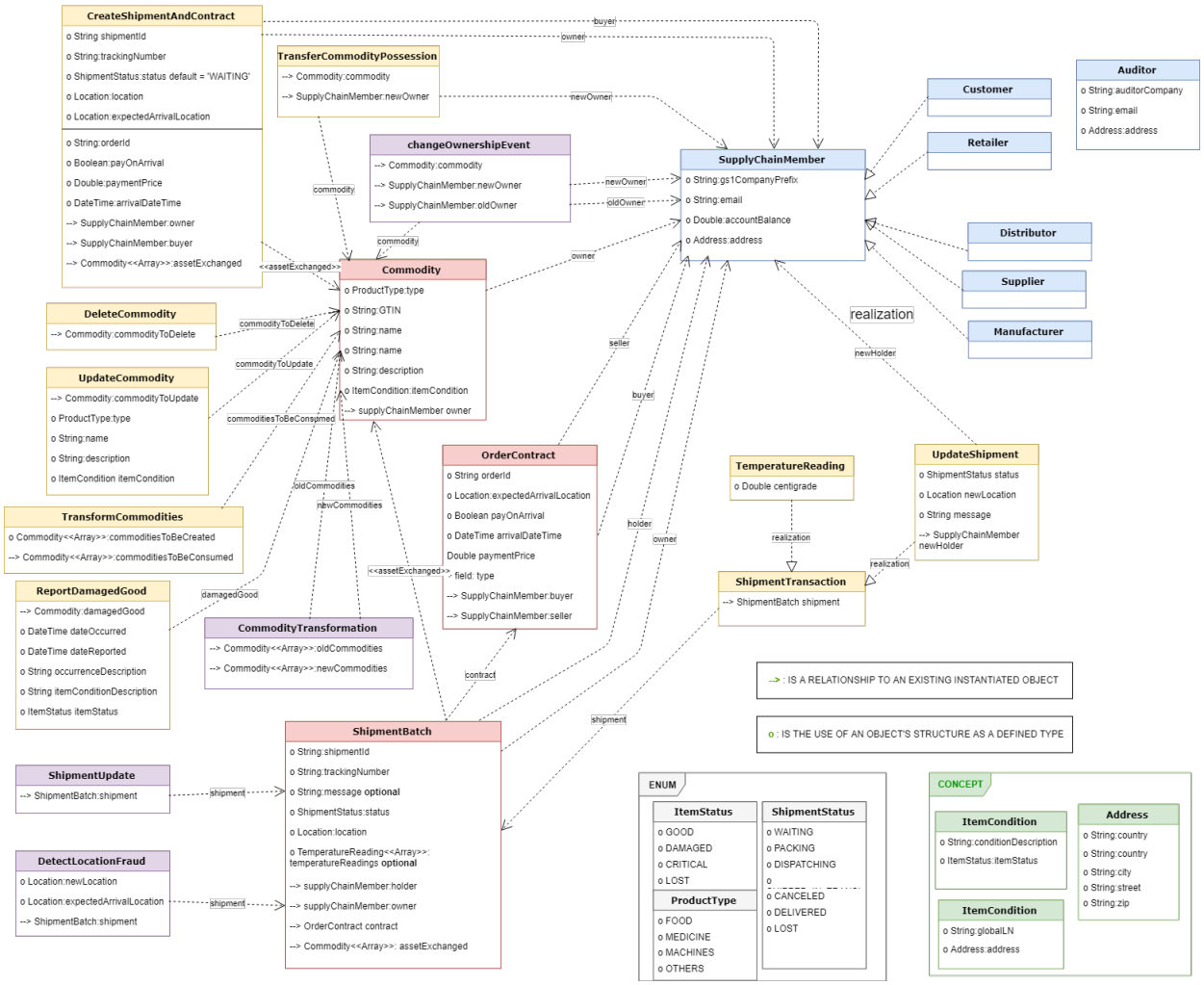


Figure B.1: Full class diagram for the designed Hyperledger Composer model

References

[1] Rhonda R Lummus and Robert J Vokurka. Defining Supply Chain Management: a Historical Perspective and Practical Guidelines. *Industrial Management & Data Systems*, 99(1):11 – 17, 2014. ISSN 10983015. doi: 10.1108/02635579910243851. Cited on pp. 1 and 2.

[2] Ram Ganeshan and Terry P Harrison. Introduction to Supply Chain Management. *Supply Chain Management An International Journal*, 47(July):3–4, 1995. ISSN 1745493X. doi: 10.1111/j.1745-493X.2011.03231.x. URL http://lcm.csa.iisc.ernet.in/scm/supply_chain_intro.html. Cited on p. 2.

[3] International Training Centre of the International Labour Organization. Understanding global supply chains, 2018. URL https://gbv.itcilo.org/index.php/briefing/show{_}paragraph/id/127.html. Date Accessed: 2018-06-27. Cited on p. 2.

[4] Ronald H. Ballou. The Evolution and Future of Logistics and Supply Chain Management. *European Business Review*, 19(4):332–348, 2007. ISSN 0955-534X. doi: 10.1108/09555340710760152. URL http://www.emeraldinsight.com/doi/10.1108/09555340710760152. Cited on p. 2.

[5] Mamun Habib. Supply chain management (scm): Theory and evolution. In Mamun Habib, editor, *Supply Chain Management*, chapter 1. IntechOpen, Rijeka, 2011. doi: 10.5772/24573. URL https://doi.org/10.5772/24573. Cited on p. 2.

[6] Alan Punter. Supply Chain Failures - A Study of the Nature, Causes and Complexity of Supply Chain Disruptions. *Airmic Technical*, pages 1–52, 2013. Cited on p. 3.

[7] Michael Prokle. Theory and Practice of Supply Chain Synchronization. *Doctoral Dissertations*, 2017. Cited on p. 3.

[8] Kari Korpela, Jukka Hallikas, and Tomi Dahlberg. Digital Supply Chain Transformation toward Blockchain Integration. pages 4182–4191, 2017. doi: 10.24251/HICSS.2017.506. URL http://hdl.handle.net/10125/41666. Cited on p. 3.

[9] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. *Www.Bitcoin.Org*, page 9, 2008. ISSN 09254560. doi: 10.1007/s10838-008-9062-0. URL https://bitcoin.org/bitcoin.pdf. Cited on p. 4.

[10] Stuart Haber and W Scott Stornetta. How to time-stamp a digital document. *Journal of Cryptology*, 3(2):99–111, jan 1991. ISSN 1432-1378. doi: 10.1007/BF00196791. URL https://doi.org/10.1007/BF00196791. Cited on p. 4.

[11] Dave Bayer, Stuart Haber, and W. Scott Stornetta. Improving the Efficiency and Reliability of Digital Time-Stamping. *Sequences II*, pages 329–334, 1992. doi: 10.1007/978-1-4613-9323-8_24. URL http://link.springer.com/10.1007/978-1-4613-9323-8{_}24. Cited on p. 4.

[12] Filiz Isik. Complexity in Supply Chains : A New Approach to Quantitative Measurement of the Supply-Chain-Complexity. *Supply Chain Management*, pages 417—-432, 2011. Cited on p. 4.

[13] Richard Wilding. The Supply Chain Complexity Triangle: Uncertainty Generation in the Supply Chain. *International Journal of Physical Distribution & Logistics Management*, 28(8):599–616, 1998. ISSN 0960-0035. doi: 10.1108/09600039810247524. URL http://www.emeraldinsight.com/doi/10.1108/09600039810247524. Cited on p. 5.

[14] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4/3:382–401, July 1982. URL https://www.microsoft.com/en-us/research/publication/byzantine-generals-problem/. Cited on p. 9.

[15] Marc Pilkington. Blockchain Technology: Principles and Applications. *Research Handbook on Digital Transformations*, pages 1–39, 2015. ISSN 1553-877X. doi: 10.4337/9781784717766.00019. URL http://papers.ssrn.com/abstract=2662660. Cited on p. 10.

[16] Vitalik Buterin. On Public and Private Blockchains, 2015. URL https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/. Date Accessed: 2018-03-20. Cited on p. 13.

[17] Vitalik Buterin. A Next-Generation Smart Contract and Decentralized Application Platform. *Ethereum White Paper*, (January):1–36, 2014. URL https://github.com/ethereum/wiki/wiki/White-Paper. Cited on p. 16.

[18] Nick Szabo. Smart Contracts: Building Blocks for Digital Markets. *URL: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/ LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html*, 1996. Cited on p. 16.

[19] Vitalik Buterin and Virgil Griffith. Casper the Friendly Finality Gadget. pages 1–10, 2017. URL https://github.com/ethereum/research/blob/master/papers/casper-basics/casper_basics.pdf. Date Accessed: 2018-02-07. Cited on p. 17.

[20] Vitalik Buterin. What Proof of Stake Is And Why It Matters, 08 2013. URL https://bitcoinmagazine.com/articles/what-proof-of-stake-is-and-why-it-matters-1377531463/. Date Accessed: 2018-03-20. Cited on p. 17.

[21] Mattias Scherer and Jerry Eriksson. Performance and Scalability of Blockchain Networks and Smart Contracts. 2017. Cited on pp. 18 and 19.

[22] Vitalik Buterin. On Sharding Blockchains. URL https://github.com/ethereum/wiki/wiki/Sharding-FAQ. Date Accessed: 2018-02-07. Cited on p. 19.

[23] Heiko Hees. What is the Raiden Network? URL https://raiden.network/101.html. Date Accessed: 2018-02-07. Cited on p. 20.

[24] Joseph Poon and Vitalik Buterin. Plasma: Scalable Autonomous Smart Contracts Scalable Multi-Party Computation. pages 1–47, 2017. URL https://plasma.io/plasma.pdf. Cited on p. 20.

[25] IBM Research. Hyperledger Fabric Architecture Explained, 2017. URL http://hyperledger-fabric.readthedocs.io/en/release-1.1/arch-deep-dive.html. Date Accessed: 2018-06-26. Cited on p. 22.

[26] IBM Research. Hyperledger Fabric Documentation. page 257, 2017. doi: 10.5281/zenodo.1009257. URL https://media.readthedocs.org/pdf/hyperledger-fabric/latest/hyperledger-fabric.pdf. Date Accessed: 2018-03-20. Cited on pp. 22 and 24.

[27] Christian Cachin. Architecture of the Hyperledger Blockchain Fabric. *IBM Research*, July, 2016. URL https://pdfs.semanticscholar.org/f852/c5f3fe649f8a17ded391df0796677a59927f.pdf. Cited on p. 22.

[28] Marko Vukolić. Hyperledger Fabric - an Open-Source Distributed Operating System for Permissioned Blockchains, 2017. URL https://blockchain-summer.epfl.ch/talks/hyperledger-fabric-vukolic.pdf. Cited on p. 22.

[29] Linux Foundation. introduction@hyperledger.github.io, 2018. URL https://hyperledger.github.io/composer/latest/introduction/introduction.html. Date Accessed: 2018-05-20. Cited on p. 25.

[30] Hyperledger Composer Development Team. Introduction @ hyperledger.github.io, 2018. URL https://hyperledger.github.io/composer/latest/introduction/introduction.html. Date Accessed: 2018-05-31. Cited on p. 26.

[31] Richard Gendal Brown, James Carlyle, Ian Grigg, and Mike Hearn. Corda: An Introduction. pages 1–15, 2016. URL https://docs.corda.net/_static/corda-introductory-whitepaper.pdf. Cited on p. 28.

[32] Martin Valenta and Philipp Sandner. Comparison of Ethereum, Hyperledger Fabric and Corda. (June):1–8, 2017. URL http://explore-ip.com/2017_Comparison-of-Ethereum-Hyperledger-Corda.pdf. Cited on p. 29.

[33] Suporn Pongnumkul, Chaiyaphum Siripanpornchana, and Suttipong Thajchayapong. Performance Analysis of Private Blockchain Platforms in Varying Workloads. *2017 26th International Conference on Computer Communications and Networks, ICCCN 2017*, 2017. doi: 10.1109/ICCCN.2017.8038517. Cited on pp. 30 and 31.

[34] Photis M. Panayides and Y. H. Venus Lun. The Impact of Trust on Innovativeness and Supply Chain Performance. *International Journal of Production Economics*, 122(1):35–46, 2009. ISSN 09255273. doi: 10.1016/j.ijpe.2008.12.025. URL http://dx.doi.org/10.1016/j.ijpe.2008.12.025. Cited on p. 34.

[35] Jeff Hoi Yan Yeung, Willem Selen, Min Zhang, and Baofeng Huo. The Effects of Trust and Coercive Power on Supplier Integration. *International Journal of Production Economics*, 120(1):66–78, 2009. ISSN 09255273. doi: 10.1016/j.ijpe.2008.07.014. URL http://dx.doi.org/10.1016/j.ijpe.2008.07.014. Cited on p. 34.

[36] CargoX. Reshaping the Future of Global Trade with World's First Blockchain Bill of Lading, 2017. URL https://cargox.io/CargoX-Whitepaper.pdf. Cited on p. 37.

[37] Juan Huertas, Hope Liu, and Sarah Robinson. Eximchain: Supply Chain Finance Solutions on a Secured Public, Permissioned Blockchain Hybrid, 2017. URL https://www.eximchain.com/EXIMCHAIN-Whitepaper.pdf. Cited on p. 38.

[38] Branimir Rakic, Tomaz Levak, Ziga Drev, Sava Savic, and Aleksandar Veljkovic. OriginTrail: First Purpose Built Protocol for Supply Chains Based on Blockchain, 2017. URL https://origintrail.io/storage/documents/OriginTrail-White-Paper.pdf. Cited on p. 39.

[39] Branimir Rakic, Tomaz Levak, Ziga Drev, Sava Savic, and Aleksandar Veljkovic. Technology @ origintrail.io, 2018. URL https://origintrail.io/technology. Date Accessed: 2018-06-07. Cited on p. 39.

[40] A.P. MOLLER - MAERSK. Maersk and IBM Launch Digital Joint Venture, 2018. URL https://www.maersk.com/stories/maersk-and-ibm-launch-digital-joint-venture. Date Accessed: 2018-02-08. Cited on p. 40.

[41] Ole Andersen and Louise Vogdrup-Schmidt. Rivals reject blockchain solution from Maersk and IBM, 2018. URL https://shippingwatch.com/carriers/Container/article10602520.ece. Date Accessed: 2018-06-07. Cited on p. 40.

[42] IEEE Computer Society. Ieee recommended practice for software requirements specifications. *IEEE Std 830-1998*, pages 1–40, Oct 1998. doi: 10.1109/IEEESTD.1998.88286. Cited on p. 81.

[43] IBM Research. Hyperledger Fabric Peers Explained, 2017. URL http://hyperledger-fabric.readthedocs.io/en/release-1.1/peers/peers.html. Date Accessed: 2018-06-27. Cited on p. 101.