University POLITEHNICA of Bucharest

Faculty of Automatic Control and Computers,
Engineering and Management of Business Systems

MASTER THESIS

Blockchain for Business

Scientific Adviser:
Dr.Ing. Gabriel Neagu

Author:
Ing. Adriana Dincă

Bucharest, 2019

I would like to thank my thesis advisor for his
support and cooperation in writing this paper.

I would also like to acknowledge
my tutor for granting me the freedom
to draw the direction of
this scientific thesis.

Abstract

Blockchain has the power to revolutionize the public and private life of every individual. Bringing decentralization and transparency in an world of informational chaos is going to change the Internet and everything else with it. The strengths of this technology are immutability - no data corruption or lost, transparency - everyone has access to any exchange, decentralization - no central authority or single point of failure. In the business world Blockchain is creating new models of business from smart contracts to open networks enterprises for a wide range of industries from finance, sharing economy to solving the prosperity paradox by eliminating the global financial exclusion. Anyway, the adoption of Blockchain is a complex process so we don't expect to see major changes in the near future.

Soon after Blockchain's white paper publication, top digital companies had visualized its impact so they started investing in research programs and professionals for adopting this technology in their business. A multi-projects collaborative effort to speed this process is the Hyperledger which aim is to simplify the requirements of developing Blockchain based applications. We have identified a business area that can be improved within the IT department of Ixia - the CapEx business procedures and proposed a hardware supply-chain based on Blockchain using two of the Hyperledger projects: Fabric and Composer. *Hardware Supply-Chain* is a system that enables its participants to track every asset manufactured and used internally by the Keysight Technologies' employees.

Contents

List of Figures

Chapter 1

Introduction

This thesis presents the Blockchain technology from a business perspective as an attempt to identify the main areas of applicability for the business world. In order to validate the information gathered in the state of the art and research stages we build a Blockchain based system in the area of hardware supply-chain for the CapEx [2] process of the NVS [3] department of ISG [4], Keysight Technologies. The content of this thesis outlines the benefits and limitations of this protocol and how the business world can revolutionize its processes by adopting it.

In this chapter we describe briefly the current state of Blockchain for business and how it can solve the biggest problems of Internet. Furthermore, we are going to outline what are our motivations for building a hardware supply-chain using Hyperledger frameworks and what objectives we want to accomplish as well as our contributions to the scientific community.

1.1 State of the art

Alex Tapscott and Don Tapscott [5] affirm that in business we need to follow the four basic principals of integrity: honesty, responsibility, transparency and consideration. Before the Blockchain innovation, the digital world relied exclusively on third party entities for certifying a business partner. The Blockchain is considered to be the only solution of the third-party issue so it is called the *Trust Protocol*. This technology has an enormous potential to revolutionize the Internet by offering it the integrity level required for business operations. Blockchain offers a new prosperity plan for peer-to-peer economy, speeding the process of financial inclusion, global protection for economic rights, eliminating bureaucracy and corruption from foreign aid, rewarding value creators, corporation redesign as the motor of capitalism, animate the IoT via collaboration, building a new Blockchain entrepreneurship, etc.

Many companies and governmental organizations consider this technology a threat because of its power to close down industries such as banks and to create a new democracy by equal distribution of decision power. In its early stage the Blockchain was associated with the black market so it offered its opponents a reason to undermine it.

However, the Blockchain's enthusiasts contend its potentially dangerous aspects and they started to build Blockchain based applications for e-commerce, electronic voting systems, digital identity, financial transaction, business collaboration, etc. For supporting the growth of Blockchain networks, the online community developed solutions to help entrepreneurs adopt

[2] Capital Expense - the budget allocated by a company to buy new assets or replace the malfunctioned ones
[3] Network Visibility Solutions
[4] Ixia Solutions Group

the protocol in their business. Hyperledger is a multi-projects organization that focuses on developing frameworks and tools to assist enterprises to integrate this protocol in their business. Fabric is the most popular project developed by Hyperledger. It offers a modular Blockchain implementation for cross-industry networks that provides an authorization layer to enable access only to trustworthy members.

1.2 Motivations and Objectives

We were motivated to choose this direction of research because it is an young technology with multiple areas of applicability for business and it offers solutions to building a new Internet where trust is embedded. As a result of this research, we reached the conclusion that systematizing the Blockchain opportunities for the business world is valuable and based on this we build some short and long term objectives:

• **Exploration stage:** analyze the existing Blockchain for business projects and identify the main industries where this protocol can revolutionize the business operations; describe the reasons why Blockchain can change the business world; outline the main characteristics of this technology that enables entrepreneurs to completely redesign their organization; systematize the business areas that can be innovated;

• **Evaluation stage:** explore Blockchain based projects to identify the best approaches to build such a system and test the most popular technologies and frameworks that offer transparency, accountability and decentralization; discover the Hyperledger community and test some of its solutions for developing Blockchain based applications from scratch without advanced technical skills; the Hyperledger organization aims to support cross-industry Blockchain networks and it offers assistance for new Blockchain ideas for enterprises;

• **Validation stage:** validate that the adoption of the Blockchain technology in business doesn't required highly-skilled professionals by using open-source assistive tools so we build a Blockchain based application using two of the Hyperledger projects: Fabric and Composer; propose a network for CapEx business operations for ISG that solves the accountability, transparency and authorization issues of the current procedure; this system is based on a business model that define the participants, their role in the CapEx process, the type of operations associated with it and a business logic for every operation involved in this procedure.

1.3 Contributions

This thesis of Blockchain for business was a tough challenge from the point of view of variety - there are a wide range of Blockchain based projects that aim to innovate the business world so the amount of information is quite vast and not everything is valid. Firstly, in chapter 2 we described the business related characteristics of Blockchain and we determine the industries where Blockchain can change the business completely. We systematized all the information gathered and presented two possible classifications: the first one is based on the industry activity and the second goes beyond the current state of business by introducing new business models and empowers everyone to prosper by creating new value or maximizing the profit by eliminating the third party.

In chapters 3 and 4 we presented the Hyperledger implementation of Blockchain and offered a brief description of the most popular projects so we claim our contribution to the Hyperledger's

promotion by offering an easy to follow guide to build Blockchain applications using Hyperledger Fabric and Composer frameworks.

Another contribution is represented by the hardware supply-chain network developed as a replacement for the CapEx process for the Network Visibility Solutions department of ISG. The network is using Hyperledger Fabric's Blockchain implementation, a simplified version of the consensus protocol and a membership service module to offer a system that is transparent - everyone can see the transactions issued in the network, participants are certified so the system offers accountability and they have limited access based on their role in the CapEx process. Therefore, we build a proof-of-concept application that solves the business operations issues of CapEx in ISG using the Blockchain protocol advantages without the need to develop a solution from scratch by integrating open-source frameworks.

1.4 Summary

The thesis contains six chapters as follows:

• **Chapter 1 - Introduction** provides a brief description about the state of the art of this research domain by presenting how the Blockchain protocol solves the business issues of the digital world. Also, in this chapter we outline the motivations and objectives of the thesis alongside our contributions to the Blockchain for business' research field.

• **Chapter 2 - Background** describes the most important characteristics of the Blockchain protocol and how Blockchain can be used to fulfill business requirements. More than that we outline the two types of Blockchain for business areas types of classification. The first type is oriented to fit the current business environment without radical changes and the second type of classification is more generic and provides insights into the future of business after embedding Blockchain - the *Trust Protocol* into digital business.

• **Chapter 3 - Study of Hyperledger Projects** aims to present the Hyperledger organization, its goals and the incubator of Blockchain based projects. Our focus is to describe the Hyperledger Fabric and Composer technical specifications in order to prepare the ground for a better understanding of the implementation aspects of Hardware Supply-Chain network.

• **Chapter 4 - Implementation** offers insights into the design and implementation details of the Hardware Supply-Chain system developed in order to outline the usefulness of this protocol in the business world. The application idea originated as a result of a release delay due to insufficient specialized hardware in one of the Network Visibility Solutions projects so we decided to solve the problems of CapEx orders' lack of transparency and no reliable time estimations for orders delivery. All these issues are solved by the Hardware Supply-Chain system that uses Hyperledger frameworks and tools.

• **Chapter 5 - Hardware Supply-Chain Network** describes the existing CapEx processes and identifies the business operations issues in the current flow of events. In order to prove how our solution fixes these problems we present the main uses case of the system in detail. In order to demonstrate the correctness and completeness of the business network model developed we use the Composer UI for testing called *Playground* [4].

• **Chapter 6 - Conclusion** builds the thesis conclusions by summarizing the thesis' contributions for Blockchain for business and suggest other research directions that weren't covered by this paper.

Chapter 2

Background

2.1 Blockchain Technology

Blockchain is a concept that offers support for a peer-to-peer electronic cash system that is able to solve the classic issues of digital money peer-based systems. The biggest challenge was the double spending issue that was solved by introducing a chain of blocks with transactions where each transaction/block has a unique hash computed using a proof-of-work algorithm. In order to add a transaction into a block it has to be validated by the majority of nodes to ensure the unique hash wasn't never added in the chain. After this validation, the transaction is added in the block and all the chains of the network update their database with this new transaction.

The proof-of-work algorithm is a hash function that requires a lot of CPU power in order to obtain a new unique hash. Thus, if an attacker tries to double spend its money it has to compute 2 hashes sequentially: the first hash is computed using the longest chain's proof-of-work and the second hash should be based on the new longest chain which already has the information about the money spend in the first transaction and as a result the network nodes are going to invalidate the second transaction. The problem still can appear if the attacker holds the majority of nodes by adding the second transaction in the chain. The possibility of having a monopoly on the system is really small if we think that it requires a lot of money to keep the nodes up and running. Additionally anyone can join the network from everywhere around the world with no real connection to the other members so this enables a wide diversity of individuals with different backgrounds and conceptions.

A node can leave the network and return to it at any time by updating its Blockchain history with the newest transactions since its inactivity so the network broadcasts the protocol messages only to the nodes that are active. When a node rejoins the network it will add the already validated transactions in its chain. The blocks' chain of transactions holds immutable information, it is the proof of all the events happened in the network so it is called the Blockchain's ledger.

The Blockchain idea was defined by Satoshi Nakamoto in his white paper called *Bitcoin: A peer-to-peer electronic cash system* [6] as a electronic payments system that uses the Bitcoin digital currency to facilitate financial transfers between peers without the involvement of a central authority. This idea solves the Byzantine Generals Problem using proof-of-work for achieving consensus. Jaroen RijnBout [7] describes how the Byzantine Generals Problem is solved using the consensus protocol proposed by Satoshi Nakamoto. He presents the story of many generals that want to crack the king's Wireless password and in order to achieve success the majority of generals must contribute to generate the CPU power at the same time to brute force the password. Thus, we need to specify a time of attack and check if the majority of

generals agree with the plan. For setting the time of attack, all the generals propose a time and compute a transaction hash. The generals can agree or not on the proposal so they can decide to add it to the ledger or not. We know that an hour of CPU power is enough to brute force the king's password and the time to compute the hash for a transaction is ten minutes if (and only if) all the generals work in the same time. Therefore, we need six generals to attack at the same time for the attempt to be successful. In order to determine if enough generals agree with the attack we need a chain of transactions that starts with the transaction that has the proposal and other six transactions added sequentially. Therefore, each general that agrees with the attack can see if the number of generals that want to attack is big enough so their attempt could succeed.

More than that, the Blockchain protocol removes the central authority by building a general solution to the Byzantin Generals Problem that allows members to reach consensus about the transactions' order. Satoshi's idea is considered the biggest revolution since the appearance of Internet and its potential goes beyond the financial world. It has the power to change the business world by innovating in the fields of supply-chain communication and provenance, cloud storage, manufacturing, electronic voting, employee payments, etc. In the following sections we are going to describe the business areas that can benefit from Blockchain and the way these changes can be accomplished along with the existing projects involved in this revolution. The technical details about the protocol are really well defined by Satoshi in his white paper [6] so our aim is to identify the main directions of applicability of the Blockchain technology with both strong and weak points.

2.2 Blockchain for Business

The Blockchain technology is an young area of research and the range of industries that can take advantages of this protocol is wide but we manage to determine the main directions of applicability. Before describing each direction it is important to briefly present the new features and concepts that were introduced after the Bitcoin public release.

The popularity of this technology has expanded rapidly and many specialists named it the *Trust Protocol*. It is the only system that solves the Third Party Issue via a fully decentralized network where each participant has equal rights of voting or mining. The process of mining refers to the process of value generating via digital currency. Each block contains a new coin transaction as the first transaction added in the block. In this way, the amount of money of the network is constantly increasing. The value of one coin is directly proportional with the amount of CPU power and time consumed to generate that coin. In a network the miners sets the fees for network's infrastructure and adjust the network traffic related parameters based on traffic analysis. In addition, there are no fees for transactions and the infrastructure fees are considerably lower than any bank or financial institution's fees.

The data stored in the ledger is encrypted so there's no way to changed it so the system can be used as a decentralized storage application.

On the other hand, the system offers transparency so all members of the network can see any transaction stored in the ledger with the mention that the author of the transaction is anonymous. All members need a public-private key to send and receive money and only the public key can be seen by the other members without any association with a real person. For public companies this level of transparency can affect their business and many of them prefer to use the classic payment system. Anyway, a transparent system offers accountability and there are areas of industries and governmental institutions that can solve the trust issue by adopting this technology.

Also, the Blockchain system is immutable so once data was added in the ledger it cannot be

changed. This feature is accomplished via a cryptographic hashing algorithm. For any data that is added in the database, a hash is computed with a fixed length. The hash has double functionality: reduces the size of any data to a fixed value and it is used by the database chain structure (a linked list) so each block has a hash pointer that references the previous block from the list and so on. In this way, only a minor change in the data will generate a completely different hash so all the previous blocks and links from the list should be regenerated. Any attempt to corrupt the ledger data has proven to be inefficient, the costs of regenerating the ledger are huge and it is quite absurd to waste CPU power in this direction while you could use it for mining.

In the first decentralized Blockchain based networks the nodes have equal rights in the process of decision. The benefit of this approach is the removal of any hierarchical structure that could undermine the wish of the majority of participants. These systems were designed to give their members equal power and to offer an alternative to a wide range of industries that use centralized systems such as governmental institutions, banks or private companies. Although these systems have a flat topology members can have different roles and their level of involvement can be different. The main limitation of this approach is related with scalability. In order to send data to all the network nodes, every node that receives a message from one of its neighbors, has to send it to all other neighbors. This mechanism is similar with gossip and has the disadvantage that it affects the network performance. As a solution to these problems, the new Blockchain based systems decided to designate some nodes to be node *leaders* that have more decision power in return to spreading the information faster to all the members of the network. This solution has the disadvantage that it is not fully decentralized but it can be a good option in situations where speed is more important than decentralization.

In 2019 the main applicability field of Blockchain remains finance. Big financial organizations started to work on projects that use this protocol. For example, US Federal Reserve is working on creating a new digital payment system that uses Blockchain and the Nasdaq stock exchange is using a Blockchain ledger - Nasdaq Linq to record shares transfers in the private market.

However, the areas where Blockchain can innovate exceed the financial world. The Blockgeeks community [8] affirms that Blockchain and the Web 3.0 will offer users the support to *create value and authenticate digital information.* Blockgeeks identifies the industries that will benefit from the adoption of Blockchain as follows:

- Smart Contracts - computer programs that define the details of a contract between network members and get executed when all the required conditions are met;
- Sharing Economy - a peer-to-peer network that enables good and services sharing without involving a third party;
- Crowd Funding - raising crowd-sourced venture capital using decentralized networks offers high transparency;
- Governance - transparent and private systems for voting and polling, smart contracts for managing administrative prerequisites and results;
- Supply-chain Auditing - Blockchain ledger can be used as a tracking and audit tool for verification about the origin and intermediaries steps from a product life-cycle;
- File Storage - a decentralized system solve the data corruption or lost; the face of Internet can be completed changed by replacing client/server web applications with decentralized systems;
- Prediction Markets - predict events using the crowd knowledge via a decentralized network;
- Intellectual Property - use smart contracts to protect copyrights and define the intellectual property market in a decentralized manner;

- Internet of Things - use smart contracts to manage the IoT systems by collecting their data and to make decisions and predictions by analyzing IoT components state;
- Microgrids - use smart contracts to sell/buy the excess of solar energy in areas close to the source point; these applications can automate the process of computing the amount of energy required and determine the cost of the energy dynamically using a IoT monitoring systems;
- Identity Management - building a global identity management system is a complex task but distributed systems can solve it; the Blockchain ledger can store also individual information such as reputation which is really important for e-commerce relationships.
- AML and KYC - refer to the anti-money laundering and know-your-customer problems that can be reduced using a cross-company ledger; it involves smaller costs for customer data analysis and creating reports of suspicious transactions;
- Data Management - building a decentralized personal data marketplace; users will collect the profit of selling their personal data to companies for data analytics and statistics.
- Land Title Registration - reduce fraud and managerial cost of registering land titles via a Blockchain ledger that guarantees immutability and transparency;
- Share Trading - improve the share arrangement process by reducing the confirmation time using a peer-to-peer network and removing any intermediaries from the arrangement;

Taking all these information into consideration, we can predict a completed different business world, where the majority makes the rules. With Blockchain we don't need banks to take care of our transactions, no more unreasonable fees on any operation. For example, a airline company can offer a flight ticketing service on Blockchain that with remove the costs associated with the online payment fees both for them and their customers. Another good example is the car sharing companies that are no longer needed if a driver can contact his clients directly in a Blockchain based network and the company payment side will go into the customer pocket. The list of use cases is huge from auction companies to all types of intermediary business. Also this technology can make some industries areas profitable again. The music singers and film makers can win more money by removal of music and streaming distributors.

Another great futurists Don Tapscott and Alex Tapscott [?] affirm in their book that *the technology likely to have the greatest impact on the future of the world economy has arrived, and it's not self-driving cars, solar energy, or artificial intelligence. It's called the blockchain.* They tried to envision the future of economy in the short, medium and long term and the results are are quite impressive. They have identified seven generic areas that would be transformed via this technology as follows:

Financial services is the first area that can be transformed by the Blockchain protocol. The principal aspects impacted by this protocol are:

- *secure cryptographic identities* that can be easily verified can interest the rating agencies, marketing and customer's data analysis organizations, regulators, etc;
- *value's transfer* without intermediaries will reduce the costs of buying/selling goods and services and this aspect can interest commercial banks, card networks, telecommunication companies and regulators;
- *value' storage* refers to new ways of storing any type of value from financial instruments, money market funds and government securities; brokerage agencies, retail and investment banks, regulators and telecommunication companies could be some of the interested parts of this transition;
- *value's loan* is any type of credit from credit cards, mortgages to governmental bonds; it can be issued, traded and paid and settled in the Blockchain ledger which will improve

the efficiency and reduce the systematic risks;

- *value's trading* enables investments banks, brokerage agencies, investors, central banks, future contracts organizations and regulators to reduce the time of transactions' settlements from days or weeks to seconds or minutes; this time reduction offers enrichment's opportunities to individual without access or with limited access to financial services;
- *funding and investing in an active* via a peer-to-peer network offer investments banks, legal and audit companies and stock exchanges the possibility to automate their dividends by smart contracts execution;
- *value's insurance and risk management* are more transparent by using decentralized markets so entities such insurance companies, risk management companies, brokerage agencies and regulators can better estimate the risks;
- *value's accounting* can be solved via smart contracts so all type of audits and accounting reporting can be generated in real time and it guarantees a high level of accuracy and transparency so regulators and audit companies can improve their activity.

Another area that can be transformed by the Blockchain technology is **the organization** by completely redesigning its architecture. The two futurists Alex and Don Tapscott describe an organizational structure that is decentralized, where managers have only a consultative role and the decisions are taken by the organization's members which are also owners. Members can work on one or more projects at a time and they get rewarded based on their contribution in the company. An interesting example is the *ConsenSys* company that is one of the pioneers of Ethereum software development. This company focuses also in building an new type of organization based on cooperation not on hierarchy where the classical jobs description is replaced with dynamic roles, the authority is distributed along the members of the organization, the rules applied are transparent and the big organizational structural changes are replaced by fast and continuous reiterations.

The **new business models** that have the potential to generate wealth are using the concept of distributed applications (dApps); starting from this idea we can identify four business models:

- *smart contracts* are a business model that has one functionality so its level of complexity is low and they are automated so there is no need for human intervention; the human support is replaced by multiple digital signatures agreements;
- *open network enterprise* is a business model with a low level of automation and a high level of technical complexity by allowing smart contracts to interact with each other and create new intelligent business decisions; in this model business is similar with networks so the organization's borders will expand;
- *autonomous agents* are a business model with a low level of automation so they require human intervention and a low level of complexity by having one functionality; in our model it represents a soft that can extract information from the environment on behalf of its owner and has the ability to make decisions independently;
- *autonomous distributed enterprise* is a business model with high levels of complexity and automation; they are the result of combining autonomous agents with open network enterprises; they are dynamic ecosystems with no hierarchy that aims to produce customer's value and owner's wealth.

The open network enterprise is a feasible business model that can replace the centralized business models that has the potential to innovate the business world and to offer better services and products at a lower cost. From this category we can identify the peer producers (members can build their reputation using the Blockchain technology both in public and private space and be rewarded accordingly), copyright owners (members can protect their intellectual property by

storing the data encrypted in the Blockchain and sell it directly), cooperatives Blockchain systems (the goods and services creators can generate more value by removing the intermediaries' side), counted/sharing economy (members can rent or sell any unnecessary goods and services), Blockchain creators (members that brings data in the ledger in order to keep track of every product from its production to the final consumer) and enterprise collaborators (members can collaborate with their knowledge to building value; for example, a social Blockchain enterprise may benefit both a user that is interested to earn money from selling some of its information such as its food preferences and for a food company that wants to launch a new type of food).

Registry of Things represents a way to share information in a distributed and secure manner, automate transactions and actions in Internet using the Blockchain technology. Its ledger is a immutable registry of all data exchanges that happen in the network so members benefit from trustworthy information. The Internet of Things can use Blockchain to store its data collections in the ledger, communicate with other IoT systems and define smart contracts to automate any agreement or payment. For example, we can define a smart contract that includes some limitations of intensive agriculture and if a farmer makes an abuse the IoT system that monitors its agricultural activity can inform authorities about the abuse.

An approach of **solving the prosperity paradox** is also connected with the Blockchain protocol. In the last decades we can observe that the gap between the rich and poor people has increased worldwide and less 1% of the global population owns more than 50% of world's wealth. This economic inequality is due to the financial exclusion: 15% of the OECD[1] countries' population never worked with a financial institution and 73% of Mexico population cannot access any banking service. For these people, the Blockchain is the only way to create a financial identity and start using banking services such as accessing loans for investments. The amount of money the Blockchain members can offer to a new financial identity can constantly increase if the member pays its debt on time and his reputation is improving.

Rebuilding the democracy and the governance is also an astringent problem of the democratic states where there is a big difference between the theory and how politicians apply the democratic principals. Many Blockchain adepts affirmed they are libertarians and anarcho-capitalists so the idea of using a decentralized system to governance over the public space and to fulfil the wish of the people wasn't really well received by the governance. By the adoption of Blockchain we can image a nation that have high performance governmental services and operations where the power is granted based on the politician ability to serve other people. Taking these ideas into consideration we can present a new historical age for democracy with electronic voting on Blockchain, new models to do politics and justice where all citizens get involved in solving their community biggest problems.

The **cultural life** is also a interesting area that can be innovated by the use of Blockchain. For example, this technology can restore musicians the possibility to distribute their music directly to their fans without paying high fees to other companies. Another use case related with this area is the intellectual property copyright so the owners can save their work encrypted on the Blockchain and sell it directly to their fans. The Blockchain can be a solution for achieving freedom of the press and freedom of expression in a world where information can be manipulated by its legal protectors due to personal fears and external pressures.

In conclusion, the Blockchain technology has the potential to revolutionize a wide range of industries and areas of day-to-day life. At this moment, it is an area of research and we aim to contribute to it by describing the state of the art of Blockchain in the business world and building a proof-of-concept using some assistive frameworks and tools for Blockchain based systems development.

[1]Organization for Economic Co-operation and Development - international organization with 36 member countries which focuses on building better policies to improve the lives of people around the world [9]

Chapter 3

Study of Hyperledger Projects

3.1 Hyperledger Organization

Hyperledger is an open source project focused on Blockchain technology that aims to bring IoT, supply chain, finance, banking and manufacturing together. Is is hosted by the Linux Foundation and includes leaders from all the mentioned areas.

The Blockchain is a peer-to-peer distributed network, each participant of the network has its own copy of the ledger, and any transaction is validated by the majority of participants. The validation is done by solving Byzantine Generals Problem via consensus. The Hyperledger project offers solutions based on Blockchain with additional features such as the *smart contracts* and it has a large number of assistive tools to facilitate the embrace of Blockchain. All projects developed under the umbrella of Hyperledger and Linux Foundation are applications that use a ledger of transactions to establish transparency, accountability and trust. These projects are for a wide variety of business areas by providing the infrastructure, tools and frameworks to build applications based on Blockchain easily and in a short period of time. Additionally these applications follow the legal constrains by keeping the network closed to authorized participants that can be made accountable for their actions.

The project was launched in 2016 under the guidance of many important corporations such as IBM, Intel, etc. The first codebases that were released to the public were Hyperledger Fabric - a product that was the work result of three organizations (OpenBlockchain from IBM, libconsensus from Blockstream and the smart contracts from Digital Asset) and Hyperledger Sawtooth from Intel. These two frameworks were offering support for growing the development of Blockchain business solutions. Later on, there were other projects released that continued to offer assistive tools to support this growth. All projects that were released under the Hyperledger project were supervised by the Hyperledger Technical Steering Committee, a group of eleven specialists that were elected from the technical contributor community. In May 2016 the Linux Foundation and the corporate members involved in Hyperledger project elected an Executive Director to make sure that this idea has all the resources needed to be successful. In this position was placed the co-founder of Apache Software Foundation - Brian Behlendorf. The project became so successful so at the end of the next year the Hyperledger counted seven more projects and the number of members increased to more that 200.

The most successful projects were Hyperledger Sawtooth - multi-language support for distributed ledger, Hyperledger Fabric - the distributed ledger written in GO language, Hyperledger Composer - framework for accelerating the process of developing Blockchain applications, Hyperledger Iroha - the distributed ledger written in C++ and Hyperledger Indy - distributed ledger for decentralized identities. All these frameworks and tools are used in real-life applica-

tions except for Hyperledger Composer which is still in incubation. The Hyperledger Composer is an assistive tool for developers to facilitate the process of building a business network using the Hyperledger Fabric. It was proven that Hyperledger Composer is hard to maintain so the Hyperledger board decided to stop the development of new features and to keep the existing functionalities compatible with the new versions of Hyperledger Fabric. The main focus of the community is to add more features to Hyperledger Fabric framework which has a modular architecture so adding new features can be done quickly and without affecting the pre-existent functionalities.

In the following sections we are going to describe the Hyperledger Fabric and Composer frameworks. The Blockchain technology promises a revolution as big as the Web and the Hyperledger group understood its potential so they formed a Blockchain incubator to offer support to any bright idea related with Blockchain technology, smart contracts and the business world.

3.2 Hyperledger Fabric

3.2.1 General Aspects

Hyperledger Fabric is an implementation of a distributed ledger developed mostly in Golang. The framework uses also other languages and technologies such as Javascript, Go or Java for smart contracts (chaincode), SDK in Node.js, Python, Java, Go and Rest. It is a solution for a secure, high-performance and permissioned Blockchain based network that has a modular architecture that allows plugins for any new features with no impact on the core functionalities.

Before getting into more details about the features provided by this framework it is important to mention that it is developed for enterprise use cases so the solution must solve the identity issue for transaction of type know-your-customer or the money laundering issue by offering transparency to authorities as participants with advanced permissions in the network. Thus this framework is permissioned which means that the participants are not anonymous, their identity is known by the other participants so they can be made accountable for their actions. The framework network is private and the transaction content is confidential so it can contain business sensitive information. In [10], Elli Androulaki and others are presenting the Hyperledger Fabric as a distributed operating system with all characteristics from the following paragraphs.

More than that the framework offers the possibility of selecting a consensus protocol based on the business needs. For example, if the business network is used by a single enterprise or it is governed by a authorized identity there's no need for using the Byzantine Fault Tolerant (BFT) protocol so it may sufficient to go with a simplified version such as Crash Fault Tolerant (CFT) consensus protocol. As a result, the Hyperledger Fabric eliminates the low latency of transaction validation and improves the network performance. The framework doesn't required CPU power for mining or for smart contract execution so the cost is similar with any distributed system.

The Hyperledger Fabric has a wide community of developers and activists that help with the development of new features and maintain the Fabric codebase. The number of contributors has grown to more than 200 members and the organizations involved to 35 so the project benefits of a diverse set of skills. With such a support the future of this project is really promising. It has modular and pluggable architecture so it can bring innovation to many industries.

3.2.2 Technical Aspects

Modularity is one of the main characteristics of Hyperledger Fabric. It was designed to have a modular architecture so it can be used by a large number of industries from manufacturing to finance.

The project was designed to be modular so all the components can be plugged in/out. The list of components (according to [11]) is formed of:

- an *ordering service* that makes sure that the order of transaction is correct and that all peers receive the blocks of transactions via broadcast; the ordering service can be plugged in/out and it runs in an independent environment (e.g Docker container named *hyperledger/fabric-orderer*);
- a *membership service provider* that enables only authorized entities to take part in the network(e.g a Docker container named *hyperledger/fabric-ca*); any entity that wants to connect to the business network has to have a cryptographic identity created by the business network Certification Authority (CA);
- *smart contracts* are applications that run independently in docker containers or other isolated environments and store agreements between network members; these smart contracts are named *chaincode* and they can be written in almost any programming language;
- a variety of *DBMS* options to store the ledger; a common option is CouchDB that runs in a Docker container named *hyperledger/fabric-couchdb*;
- more application's constrains for approval and validation policy;
- an optional *peer-to-peer gossip service* to leverage messages to all the peers by broadcast; makes sure that everyone receives up-to-date information about the ledger transactions and fills in any gaps if one node missed some of the broadcasts.

The chaincode (smart contract) is an application that facilitates and verifies the execution of a digital agreement between one or more network participants. The ledger is responsible for executing the digital contract at the time that it need to be applied. In a business network one or more digital contracts can co-exist and everyone that is connected to that network can create a contract. The network is responsible for validating that application code and it should be consider invalid until it is validated using order-execute architecture. In most Blockchain systems the smart contracts follow the rule order-execute so to reach consensus the order-execute design should be deterministic. Building a deterministic architecture is quite challenging and the code should be written in domain-specific programming languages (e.g Ethereum developed a DSL - Solidity for their decentralized applications) in most of the cases. Additionally this architecture brings performance issues due to the fact that transactions are added sequentially in the ledger.

The Hyperledger Fabric tries to solve the overhead of using the order-execute approach by proposing a new architecture type - execute-order-validate. This new design is composed by three stages. In the first stage the transactions are checked for correctness and if so they are immediately executed. After that, they are ordered using a consensus protocol and in the last stage they are validated according to an endorsement policy and added to the ledger. As a result of this approach the Hyperledger Fabric solves the issue of using DSL and non-determinism by removing any inconsistency and reduces the performace overhead. By removing any non-deterministic problem, the smart contracts can be written in any standard programming language. The first supported languages are Go and Node.js but the board of Hyperledger plans to support more popular programming languages such as Java in the future releases.

Permissioned Blockchain refers to restricting the access to the network to a known group of participants that are worth it to be trusted. It is not necessary that participants may know

each other but all of them should be vetted by a trustful authority. The Fabric is using a permissioned Blockchain system so each member of the business network can easily be identified and if his/her actions (e.g deploying a smart contract or changing the configuration of the network) are malicious that member can be made accountable for them. In a permissionless Blockchain system it is very hard to identify the real person that stood behind a certain action so Hyperledger Fabric is more suitable for the business world. The Certification Authority is one instrument that can vet for the business network members and any respectful business can easily obtain the authorization to participate in the network.

Privacy and confidentiality is key for most of the business use-cases therefore any solution used has to guarantee that the data cannot be available to the public. In permissionless systems the transactions' data is available to every node in the Blockchain network so such a system fails to fulfill the privacy business requirements. The main approach of permissionless Blockchain systems is to encrypt their data. However, this solution is far from perfect because other organizations can hold enough computational power to break any encryption in a reasonable amount of time. The approach of permissioned systems is to restrict the distribution of private data to only the involved participants via a channel. The Hyperledger Fabric allows any member of the network to create channels for different purposes so they can share confidential data with the authorized members. Another solution that can solve this issue for both permissionless and permissioned Blockchains is the Zero Knowledge Proofs (ZKP) protocol but it is still under research so in the meantime creating a channel to share data with authorized entities is a good alternative.

As mentioned above, the Hyperledger Fabric offers support for pluggable components for many of its features. The ordering service is a pluggable module that is responsible of achieving consensus of ordering transactions. In the literature, there are two popular consensus protocols: Crash Fault Tolerant (CFT) and Byzantine Fault Tolerant (BFT). Each of them can have one or more implementations. For example, the Hyperledger Fabric has two possible implementations for the CFT algorithm: Kafka (based on Zookeeper) and *etcd* library from Raft.

The performance of Hyperledger Fabric depends on a variety of parameters: the number of nodes of the network, the number and size of transactions and hardware limitations so it is difficult to generalize the performance of the Hyperledger Fabric project. In order to determine the performance of a Business Network based on Fabric, the Hyperledger team has developed another project called Hyperledger Caliper. The community is contributing to define the set of measures that affect performance and scalability to help Caliper team build a powerful framework for benchmarking.

Taking all these into account, the Hyperledger Fabric is one of the best solution for permissioned distributed systems that integrates the Blockchain technology. The Hyperledger community is really involved in improving and building strong solutions for Blockchain business networks and the Hyperledger Fabric project has the biggest support from the community. Fabric supports a wide range of industries: from banking, manufacturing, supply chain to retail, healthcare, etc.

3.3 Hyperledger Composer

The Hyperledger Composer framework is one of the assistive tools developed by Hyperledger team that focuses on helping developers accelerate the process of creating Blockchain systems for business. Building Blockchain applications from scratch require high technical skills from networking protocols, cryptography and consensus algorithms and the development period can extend to one or more years. Having these issues in mind, Hyperledger community started a project to reduce the time and the skills set required to build Blockchain based systems. The project is called Hyperledger Composer and it is really appreciated by developers by reducing the time and effort from years to months.

The Hyperledger Composer framework can be easily integrated with Fabric via a Loopback REST API. Before getting into more details on how we can integrate the two projects it is important to describe the idea behind Hyperledger Composer.

This framework is using a Business Network Archive (BNA) to store information about the participants of the Blockchain network and the assets they can store in the ledger via transactions. Each business requires a custom BNA which defines the type of members, their roles and permissions and the transactions that can be done in the Blockchain. For example, in the supply-chain industry we need one or many manufacturers, buyers/clients, distributors and regulators. All these participants have restricted access to submit transactions or change the network configurations based on their role. For example a buyer can submit an *order* transaction but he/she cannot submit a *delivery* transaction.

The BNA has a well defined structure and each component should be defined using a custom Composer syntax. According to Hyperledger Composer Community [12] the BNA has to follow this structure:

- one or more Model files - written using Hyperledger Composer Modeling Language, an object oriented language that enables developers to define three elements: the namespace (only one for all the resources declared in the model), the list of entities of the network: participants, assets, events and transactions and the possibility to add entities defined in other namespaces;
- one or more Transactions Functionalities file - each transaction should have a metadata, decorators and a JavaScript function; these functions run when a participant submits a transaction; the REST API exposed by Hyperledger Fabric is invoked in the transaction function via *getNativeAPI* and the Hyperledger Composer API via *getAssetRegistry*; also these functions can issue events that other external systems can listen to via the *composer-client* library;
- an Access Control role file - written using Hyperledger Composer Access Control language that enables setting permissions for all the entities defined in the model; the rules that can be enabled/disabled are Create Read Update and Delete (C.R.U.D.); there are two types of access controls: business and network; both types are defined in the Access Control files and the business control is applied after the network control;
- a Query Definition file - it is an optional component that requires the *composer-rest-server* to be up and running; there are two types of queries: named and dynamic; the named queries are static, defined in the BNM and exposed by the REST server via GET endpoints; the dynamic queries can be defined in transaction functions or directly in the client application.

The Figure 3.1 outlines the BNA structure with all the components and a brief description for each of them. .

There two possible ways to use this framework: the playground and the CLI tools. The first solution offers a web user-interface to define, deploy and test business network directly on cloud or locally on the personal machine. The other solution is for Blockchain developers that want access to all the Fabric functionalities and that can be easily integrated with Visual Studio Code IDE and Atom editor.

More than that, the Hyperledger Composer has a specialized storage mechanism which decouples transactions information from any other information such as participants and assets data. All transactions are stored in the Blockchain ledger, everything else is located in the state database associated with that ledger. Another interesting feature is the Historian Registry that stores the transactions as *HistorianRecord* assets of the business network. These assets' definition is available in the Hyperledger Composer Model and they store information about the

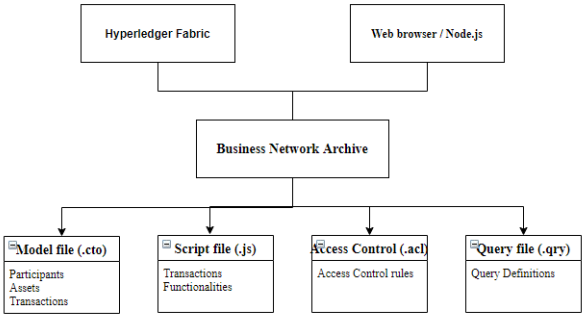

Figure 3.1: Hyperledger Composer Business Network Archive (inspired by [1])

transaction that created them, the identity of the participant and the participant that invoked the transaction.

Another concept related with the Composer framework is the Connection Profile. Each Blockchain Network has a Connection Profile that defines the system and it is created by the network administrator to enable members to connect to the system. The Hyperledger Composer uses the Identity concept to ensure that only authorized members connect to the network. An Identity has a private key, a digital certification and a metadata file that is optional and may store the network name and other details about it. A Connection Profile and an Identity form a Business Network Card that offers a member access to the network. It is worth mentioning that only one identity can be stored in a Business Network Card and that identity may be associated with one of the participant that exists in the network. The concept of Business Network Cards was introduced to enable groups of participants with the same access rights to access the network so it is related with the network access control management.

Chapter 4

Implementation

4.1 General Aspects

After analyzing the existing Blockchain solutions we reached the conclusion that Hyperledger Fabric and Composer are suitable for the enterprise usage and we propose an application build using these two frameworks that solves the CapEx [2] related issues in the division ISG [3] of Keysight Technologies Inc. This proof of concept focuses on hardware supply for the research and development teams of NVS [4] department.

Many software applications developed by ISG run on dedicated hardware and in many situations only one employee can use that hardware at a time. The existing CapEx procedure is very complicated and the time between ordering a new device and receiving it can be really long. It may happen that it isn't needed anymore by the time the device is received. The dedicated hardware is expensive (some devices may cost up to $50000) so not receiving it on time may cost the department a lot of money and it can also delay the releases.

On each quarter, the manager has the responsibility of determining what are the hardware requirements of his teams and send the list to her/his superior for approval. Also, the manager has to check the price of the requested devices and if the costs are over the CapEx budget, the hardware list should be prioritized. After completing the list and setting the priorities, the manager has to fill in an *Internal Sales Order*(ISO) on an ordering platform. After creating the ISO the manager has to wait for the order to be delivered.

A huge disadvantage of this procedure is the lack of transparency regarding the status of the order and the time estimation until receiving the order. In addition to these problems, the costs may differ due to the international tax changes.

The project idea is to use the Blockchain strengths to solve the transparency problem so any authorized member of the network can check the status of a CapEx order and it also improves the overall time of order processing. The Blockchain protocol uses proof-of-work to guarantee that only valid blocks of transactions are added in the ledger so once that data is added there's no way to modify it without being invalidated by the other members. As a result, all the orders added in the CapEx supply chain are visible to all the authorized members so any attempt of crime is a failure.

The CapEx procedure involves also some difficult and time consuming discussions to get the approval for a certain CapEx order. The employees that order a hardware device need to get

[2] Capital Expense
[3] Ixia Solution Group
[4] Network Visibility Solutions

approval from their manager and the manager must check the CapEx budget before approving it. All these issues can be easily managed in a permissioned blockchain network where the participants have different levels of access based on their roles. This type of Blockchain solves the accountability issue by allowing access to the network only to authorized members.

After some research work, we discovered that Hyperledger Composer is a good solution to limit the network access based on the participant role in the business. This framework offers two layers of access control: network and business. The network layer is the first level of access so each participant has to use a unique identification network card to connect to the Blockchain network. The second level of access control is the business layer that restrict participants to do certain actions based on their role in the business. For example, a participant that can order a hardware doesn't have access to schedule it for manufacturing.

Taking all of these into account, we build a project that uses Hyperledger Composer for business modeling, Hyperledger Fabric for accessing the Blockchain protocol via an API and solves all the issues described above.

4.2 Business Network

In order to develop the hardware supply chain using the Blockchain protocol without too much effort we choose the frameworks implemented by Hyperledger that facilitate the adoption of the Blockchain technology. The usage of these frameworks reduce the time and effort to develop the supply chain consistently.

The BNM [1] is a concept introduced by the Hyperledger Composer project that facilitates the access to all the Fabric features such as introducing members with different roles or developing business functionalities in JavaScript language. For defining the business logic we can use the Hyperledger Composer API to connect to the Fabric or invoke the Fabric native functions. In most of the enterprise use cases we have to provide a mechanism to restrict the network access and to have a transparent view of everything that happens in the system. These two frameworks fulfill really well the requirements of the business world so it is the obvious solution to select them for any Blockchain for business project.

The project idea is to handle the CapEx procedures so it has to follow a certain business model. In the NVS department, a CapEx process involves many actors from team leaders, managers, manufactures and external authorities. Keysight Technologies is an international company and it has to make sure that all the processes conducted internally and externally are complied with the laws where it operates.

In the first stage we focused on defining a supply-chain solution that handles only the CapEx processes conducted internally due to a lack of visibility about the external procedures of the delivery stage. Therefore we have identified three types of participants: *Employee*, *Manufacturer* and *Regulator*, the assets traded are the *Order* and *Chassis* and the type of transactions submitted are *PlaceOrder* and *UpdateOrderStatus*.

As described in section 3.3 we developed a BNA [2] that follows the structure required by the Hyperledger Composer framework (see Figure 4.1). In the following paragraphs we are going to get into the implementation aspects of the supply-chain project in order to outline the advantages of integrating the Hyperledger frameworks and tools in the CapEx business process.

The Hardware Supply-Chain's business model was inspired by the *Vehicle Manufacture Network* sample offered by Hyperledger Composer and its source code can be found on Hyperledger Composer github repository [3].

[1] Business Network Model
[2] Business Network Archive
[3] https://github.com/hyperledger/composer-sample-networks.git

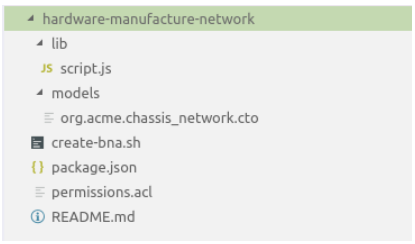

Figure 4.1: Structure of the *hardware-manufacture-network.bna* file

The Business Network Model is defined in *org.acme.chassis_network.cto* file and contains information about participants, assets, transactions, events, enums and concepts. All of these refer to the network resources and require to be associated with a namespace. The namespace used by all these resources is *org.acme.chassis_network*. A BNA can import resources from other namespaces and this practice is used when more than one organization is connected to the network. In our case there's only one organization that has access to the network and there's no need for other namespaces.

The first group of resources is the class definition group and it includes the class types of Participant, Asset and Transaction. In the supply chain we have three types of participants so there are four classes definition of type participant: *Employee*, *Division* and its subclasses *Manufacturer* and *Regulator*, two types of assets so there are two classes definition of type asset: *Chassis* and *Order* and two types of transactions so there are two classes definition of type transaction: *PlaceOrder* and *UpdateOrderStatus*.

In the composer modeling language, classes of type *Participant* must be defined using the keyword *participant* as it can be seen in Figure 4.2. Each participant type requires an identification property that is set using the phrase *identified by* after the class name. For example, the *Employee* participant is identified using its identification number called *employeeId* and has a set of properties defined after the *o* syntax. This modeling language is object-oriented and we can make use of the inheritance property so both the *Manufacturer* and *Regulator* are subclasses of the participant *Division*. A participant of type *Regulator* is someone from the division's management team (e.g a project manager).

```
15 participant Employee identified by employeeId {
16   o String employeeId
17   o String name
18   o String email
19 }
20
21 abstract participant Division identified by name {
22   o String name
23 }
24
25 participant Manufacturer extends Division {
26 }
27
28 participant Regulator extends Division {
29 }
```

Figure 4.2: Hardware Supply-Chain: Participants (inspired by [2])

Another type of resources are the assets. In our business network we have identified two types of assets: *Order* and *Chassis*. An asset is defined using the reserved keyword *asset* followed by the asset's name, the phrase *identified by* and the name of the variable used for identification. The *Order* asset is used to start the ordering procedure and have a mechanism to track all the intermediate steps previous to register the new hardware device. The *Chassis* asset is used to track the hardware devices that were registered. In the Keysight hardware divisions, the chassis is identified using a serial number. The other properties of this asset are the *chassisDetails* - a

field of type concept (which is an abstract class with no identification key), the *chassisStatus* - an enum that stores information about the life-cycle stage of a device (never used, active, returned and scrapped) and the *order* - the employee that submitted the order (it is referenced with a reserved syntax –>). In Figure 4.3 and Figure 4.4 are defined the two type of asset resources used by the supply-chain.

```
44  concept ChassisDetails {
45    --> Manufacturer make
46    o String chassisType
47    o Integer noLineCards
48    o String processorType
49    o Integer noProcessors
50    o Integer noCore
51  }
52
53  asset Chassis identified by serialNumber {
54    o String serialNumber
55    o ChassisDetails chassisDetails
56    o ChassisStatus chassisStatus
57    --> Employee owner optional
58  }
59
60  enum ChassisStatus {
61    o ACTIVE
62    o OFF_THE_ROAD
63    o SCRAPPED
64    o RETURN
65  }
```

Figure 4.3: Hardware Supply-Chain: Asset of type *Chassis* (inspired by [2])

The model has to contain also the type of transactions that can be submitted in the network and the events connected with them. In our business model we determined the need of two type of transactions: *PlaceOrder* and *UpdateOrderStatus*. These two types are required in order to have a complete flow of the CapEx ordering process. In the first stage an employee needs to place an order, the order is than approved by the regulator and send to the manufacturer. During manufacturing, the order status is changed from scheduled for manufacturing to serial number assigned and then delivered. In the CapEx procedures the manager is responsible to receive the orders and assign an owner for each of the devices. In Figure 4.5 and Figure 4.6 we outline the transactions definitions and the events associated: *PlaceOrderEvent* and *UpdateOrderStatusEvent*.

Each participant has a network card to connect to the supply-chain system that is provided by an administrator. In addition to the network access control we can restrict the actions done by a member of the network by defining rules for all the possible actions. The possible type of operations are create, read, update and delete (CRUD) and we can use this operations to create rules for all types of participants in report with other resources.

In the supply-chain network the *Employee* has to follow three rules: *EmployeeMakeOrder*, *EmployeePlaceOrder* and *EmployeeReadOrder* (see Figure 4.7). In these rules the participant is *org.acme.chassis_network.Employee*, the resources used are *org.acme.chassis_network.PlaceOrder* and *org.acme.chassis_network.Order* and the rules specify if it is allowed or not to perform one

```
asset Order identified by orderId {
  o String orderId
  o ChassisDetails chassisDetails
  o OrderStatus orderStatus
  o Options options
  --> Employee orderer
}

concept Options {
  o String comment
  o String returnComment
}

enum OrderStatus {
  o PLACED
  o SCHEDULED_FOR_MANUFACTURE
  o SERIAL_NUMBER_ASSIGNED
  o OWNER_ASSIGNED
  o NOT_IN_ACCORDANCE
  o DELIVERED
}

concept ChassisDetails {
  --> Manufacturer make
  o String chassisType
  o Integer noLineCards
  o String processorType
  o Integer noProcessors
  o Integer noCore
}
```

Figure 4.4: Hardware Supply-Chain: Asset of type *Order* (inspired by [2])

of the operations of CRUD.

For the participant *Manufacturer* we had defined rules that involves the asset of type *Chassis* (see Figure 4.8) and the asset of type *Order* (see Figure 4.9). A participant of type *Manufacturer* can create new chassis and view the existing devices manufactured in the hardware department. Also this type of participant can update the status of order and read the existing orders assigned to him/her.

The last rule is associated with a participant of type *Regulator* who has extended permissions in the network. He/She can see all the orders submitted, who is the employee that ordered it and has the possibility to assign a chassis to one of the employees (see Figure 4.10). This type of participant has a managerial role in the division so he/she is authorized to make changes in the network, to validate or invalidate a certain order and track the cause of orders' delays or other issues.

Additionally to all of these, we need to set some default rules for the system administrator that is responsible with network and business access control. The admin can create network cards for new members or deploy a new version of the *.bna*. We can also set some generally applied

```
67  transaction PlaceOrder {
68  |  o String orderId
69  |  o ChassisDetails chassisDetails
70  |  o Options options
71  |  --> Employee orderer
72  }
73
74  event PlaceOrderEvent {
75  |  o String orderId
76  |  o ChassisDetails chassisDetails
77  |  o Options options
78  |  --> Employee orderer
79  }
```

Figure 4.5: Hardware Supply-Chain: Transaction & Event *PlaceOrder* (inspired by [2])

```
81  transaction UpdateOrderStatus {
82  |  o OrderStatus orderStatus
83  |  o String serialNumber optional
84  |  --> Order order
85  }
86
87  event UpdateOrderStatusEvent {
88  |  o OrderStatus orderStatus
89  |  o Order order
90  }
```

Figure 4.6: Hardware Supply-Chain: Transaction & Event *UpdateOrderStatus* (inspired by [2])

rules for the network members. The rules of the system administrator and network members are described in Figure 4.11. All participants can see the other members of the network. We think this rule is really useful in the early days of the network or for new members to learn about the system participants and be able to connect to them if they are in a situation that requires information from someone else in the business network.

The Hyperledger Composer has defined a default namespace *org.hyperledger.composer.system* that has some predefined participants that can be used to enable the network administrator special privileges. In our business, all participants have access to the system resources and only the *org.hyperledger.composer.system.NetworkAdmin* has full access to the user resources.

Another important aspect of the project implementation is the transactions definition. Transactions hold all the business logic and they use the Hyperledger APIs to connect to the Fabric and register the submitted transactions in the ledger.

For each transaction we have to define a function that holds the business logic. The *placeOrder* function (see Figure 4.12) is executed every time a transaction of type *PlaceOrder* is submitted. In this function we used some of the Hyperledger Composer API calls. The list of API calls used is:

- *getFactory* - returns the *module:composer-runtime.Factory* which is used to create, update resources and to create relationships between resources;

```
4   rule EmployeeMakeOrder {
5       description: "Allow Employees to create and view orders"
6       participant(e): "org.acme.chassis_network.Employee"
7       operation: CREATE
8       resource(o): "org.acme.chassis_network.Order"
9       transaction(tx): "org.acme.chassis_network.PlaceOrder"
10      condition: (o.orderer.getIdentifier() == e.getIdentifier())
11      action: ALLOW
12  }
13
14  rule EmployeePlaceOrder {
15      description: "Allow Employees to place orders and view they've done this"
16      participant(e): "org.acme.chassis_network.Employee"
17      operation: CREATE, READ
18      resource(o): "org.acme.chassis_network.PlaceOrder"
19      condition: (o.orderer.getIdentifier() == e.getIdentifier())
20      action: ALLOW
21  }
22
23  rule EmployeeReadOrder {
24      description: "Allow Employees to place orders and view they've done this"
25      participant(e): "org.acme.chassis_network.Employee"
26      operation: READ
27      resource(o): "org.acme.chassis_network.Order"
28      condition: (o.orderer.getIdentifier() == e.getIdentifier())
29      action: ALLOW
30  }
```

Figure 4.7: Hardware Supply-Chain: Employee Rules (inspired by [2])

- *factory.newResource* - creates an asset;
- *factory.newRelationship* - creates a relationship between two resources;
- *factory.newEvent* -creates an event for the transaction;
- *emit* - sends an event for the transaction;
- *getAssetRegistry* - returns a *Promise* and it tries to get the asset registry with the specified identification name; if the registry exists the call return *@link module:composer-runtime.AssetRegistry AssetRegistry*, otherwise it returns an error.

The *placeOrder* function creates an asset of type *Order*, register it in the database state and emits an event for this transaction.

The second function called *updateOrderStatus* is executed when a member submits a transaction

```
60  rule ManufacturerCreateChassis {
61      description: "Allow manufacturers to create and view their chassis"
62      participant(m): "org.acme.chassis_network.Manufacturer"
63      operation: CREATE
64      resource(c): "org.acme.chassis_network.Chassis"
65      transaction(tx): "org.acme.chassis_network.UpdateOrderStatus"
66      condition: (c.chassisDetails.make.getIdentifier() == m.getIdentifier() && tx.orderStatus == "SERIAL_NUMBER_ASSIGNED")
67      action: ALLOW
68  }
69
70  rule ManufacturerReadChassis {
71      description: "Allow manufacturers to create and view their chassis"
72      participant(m): "org.acme.chassis_network.Manufacturer"
73      operation: READ
74      resource(c): "org.acme.chassis_network.Chassis"
75      condition: (c.chassisDetails.make.getIdentifier() == m.getIdentifier())
76      action: ALLOW
77  }
```

Figure 4.8: Hardware Supply-Chain: Manufacturer Chassis related Rules (inspired by [2])

```
rule ManufacturerUpdateOrder {
    description: "Allow manufacturers to view and update their own orders"
    participant(m): "org.acme.chassis_network.Manufacturer"
    operation: UPDATE
    resource(o): "org.acme.chassis_network.Order"
    transaction(tx): "org.acme.chassis_network.UpdateOrderStatus"
    condition: (o.chassisDetails.make.getIdentifier() == m.getIdentifier())
    action: ALLOW
}

rule ManufacturerUpdateOrderStatus {
    description: "Allow manufacturers to update order statuses and view they've done this"
    participant(m): "org.acme.chassis_network.Manufacturer"
    operation: CREATE, READ
    resource(o): "org.acme.chassis_network.UpdateOrderStatus"
    condition: (o.order.chassisDetails.make.getIdentifier() == m.getIdentifier())
    action: ALLOW
}

rule ManufacturerReadOrder {
    description: "Allow manufacturers to view and update their own orders"
    participant(m): "org.acme.chassis_network.Manufacturer"
    operation: READ
    resource(o): "org.acme.chassis_network.Order"
    condition: (o.chassisDetails.make.getIdentifier() == m.getIdentifier())
    action: ALLOW
}
```

Figure 4.9: Hardware Supply-Chain: Manufacturer Order related Rules (inspired by [2])

```
rule RegulatorAdminUser {
    description: "Let the regulator do anything"
    participant: "org.acme.chassis_network.Regulator"
    operation: ALL
    resource: "**"
    action: ALLOW
}
```

Figure 4.10: Hardware Supply-Chain: Regulator Rule (inspired by [2])

of type *UpdateOrderStatus*. We used the same API calls to interact with the Hyperledger Composer and Fabric frameworks. This function has a more complex business logic. In the case a member changes the order status to *SERIAL_NUMBER_ASSIGNED* a new asset of type *Chassis* is created and registered. If the order status is changed to *OWNER_ASSIGNED* the chassis is assigned to an owner of type *Employee* and the chassis status is set to *ACTIVE*. In all cases, we update the order status with the new value (see Figure 4.13).

In order to start using the business network we need to deploy the BNA on Hyperledger Fabric and to create a network admin card to connect to it. The Composer community offers two possibles solutions for testing the defined BNA so we can use the *Playground* UI or a REST Server which enables users to connect to the network using an API. The Playground is suitable for proof-of-concept projects and it can be used from browser directly or installed locally. The REST Server is used for production and it generates REST API based on the BNA content so any client application can connect to the server via REST endpoints.

We choose to deploy and test our BNA using the Composer REST Server [3]. The commands run to install and test the BNA can be found in *start-rest-server.sh* script (see A.1).

All endpoints generated by Composer REST Server were documented using Swagger [1].

Therefore, the Hardware Supply-Chain participants can interact with our BNA via REST so

[1] Swagger - framework used to design, build, document and test RESTful Web Services

```
87  rule ParticipantsSeeSelves {
88      description: "Let participants see themselves"
89      participant(e): "org.hyperledger.composer.system.Participant"
90      operation: ALL
91      resource(r): "org.hyperledger.composer.system.Participant"
92      condition: (r.getIdentifier() == e.getIdentifier())
93      action: ALLOW
94  }
95
96  rule NetworkAdminUser {
97      description: "Grant business network administrators full access to user resources"
98      participant: "org.hyperledger.composer.system.NetworkAdmin"
99      operation: ALL
100     resource: "**"
101     action: ALLOW
102 }
103
104 rule System {
105     description: "Grant all full access to system resources"
106     participant: "org.**"
107     operation: ALL
108     resource: "org.hyperledger.composer.system.**"
109     action: ALLOW
110 }
```

Figure 4.11: Hardware Supply-Chain: Predefined Participants Rules (inspired by [2])

any web client should be able to consume it easily. In Figure 4.14 we can see all the endpoints generated from our BNA. The endpoints base url is */api* and the root path is dynamically generated when starting the *composer-rest-server* (usually it runs on *localhost:3000*). Each endpoint can be tested using the Composer REST Server Explorer based on Swagger. (see Figure 4.15)

```
/**Place an order for a chassis
 * @param {org.acme.chassis_network.PlaceOrder} placeOrder - the PlaceOrder transaction
 * @transaction */
async function placeOrder(orderRequest) { // eslint-disable-line no-unused-vars
    console.log('placeOrder');

    const factory = getFactory();
    const namespace = 'org.acme.chassis_network';

    const order = factory.newResource(namespace, 'Order', orderRequest.orderId);
    order.chassisDetails = orderRequest.chassisDetails;
    order.orderStatus = 'PLACED';
    order.orderer = factory.
        newRelationship(namespace, 'Employee', orderRequest.orderer.getIdentifier());
    order.options = orderRequest.options;

    // save the order
    const assetRegistry = await getAssetRegistry(order.getFullyQualifiedType());
    await assetRegistry.add(order);

    // emit the event
    const placeOrderEvent = factory.newEvent(namespace, 'PlaceOrderEvent');
    placeOrderEvent.orderId = order.orderId;
    placeOrderEvent.chassisDetails = order.chassisDetails;
    placeOrderEvent.options = order.options;
    placeOrderEvent.orderer = order.orderer;
    emit(placeOrderEvent);
}
```

Figure 4.12: *PlaceOrder* Function (inspired by [2])

```
/**Update the status of an order
 * @param {org.acme.chassis_network.UpdateOrderStatus} updateOrderStatus - the UpdateOrderStatus trans
 * @transaction */
async function updateOrderStatus(updateOrderRequest) { // eslint-disable-line no-unused-vars
    console.log('updateOrderStatus');
    const factory = getFactory();
    const namespace = 'org.acme.chassis_network';

    // get chassis registry
    const chassisRegistry = await getAssetRegistry(namespace + '.Chassis');
    if (updateOrderRequest.orderStatus === 'SERIAL_NUMBER_ASSIGNED') {
        if (!updateOrderRequest.serialNumber) {
            throw new Error('Value for serialNumber was expected');
        }
        // create a chassis
        const chassis = factory.
            newResource(namespace, 'Chassis', updateOrderRequest.serialNumber );
        chassis.chassisDetails = updateOrderRequest.order.chassisDetails;
        chassis.chassisStatus = 'OFF_THE_ROAD';
        await chassisRegistry.add(chassis);
    } else if(updateOrderRequest.orderStatus === 'OWNER_ASSIGNED') {
        if (!updateOrderRequest.serialNumber) {
            throw new Error('Value for serialNumber was expected');
        }

        // assign the owner of the chassis to be the Employee who placed the order
        const chassis = await chassisRegistry.get(updateOrderRequest.serialNumber);
        chassis.chassisStatus = 'ACTIVE';
        chassis.owner = factory.
            newRelationship(namespace, 'Employee', updateOrderRequest.order.orderer.employeeId);
        await chassisRegistry.update(chassis);
    } else if(updateOrderRequest.orderStatus === 'NOT_IN_ACCORDANCE') {
        if (!updateOrderRequest.serialNumber) {
            throw new Error('Value for serialNumber was expected');
        }
        const chassis = await chassisRegistry.get(updateOrderRequest.serialNumber);
        chassis.chassisStatus = 'RETURN';
        chassis.owner = null;
        await chassisRegistry.update(chassis);
    }
    // update the order
    const order = updateOrderRequest.order;
    order.orderStatus = updateOrderRequest.orderStatus;
    const orderRegistry = await getAssetRegistry(namespace + '.Order');
    await orderRegistry.update(order);
    // emit the event
    const updateOrderStatusEvent = factory.newEvent(namespace, 'UpdateOrderStatusEvent');
    updateOrderStatusEvent.orderStatus = updateOrderRequest.order.orderStatus;
    updateOrderStatusEvent.order = updateOrderRequest.order;
    emit(updateOrderStatusEvent);
}
```

Figure 4.13: *UpdateOrderStatus* Function (inspired by [2])

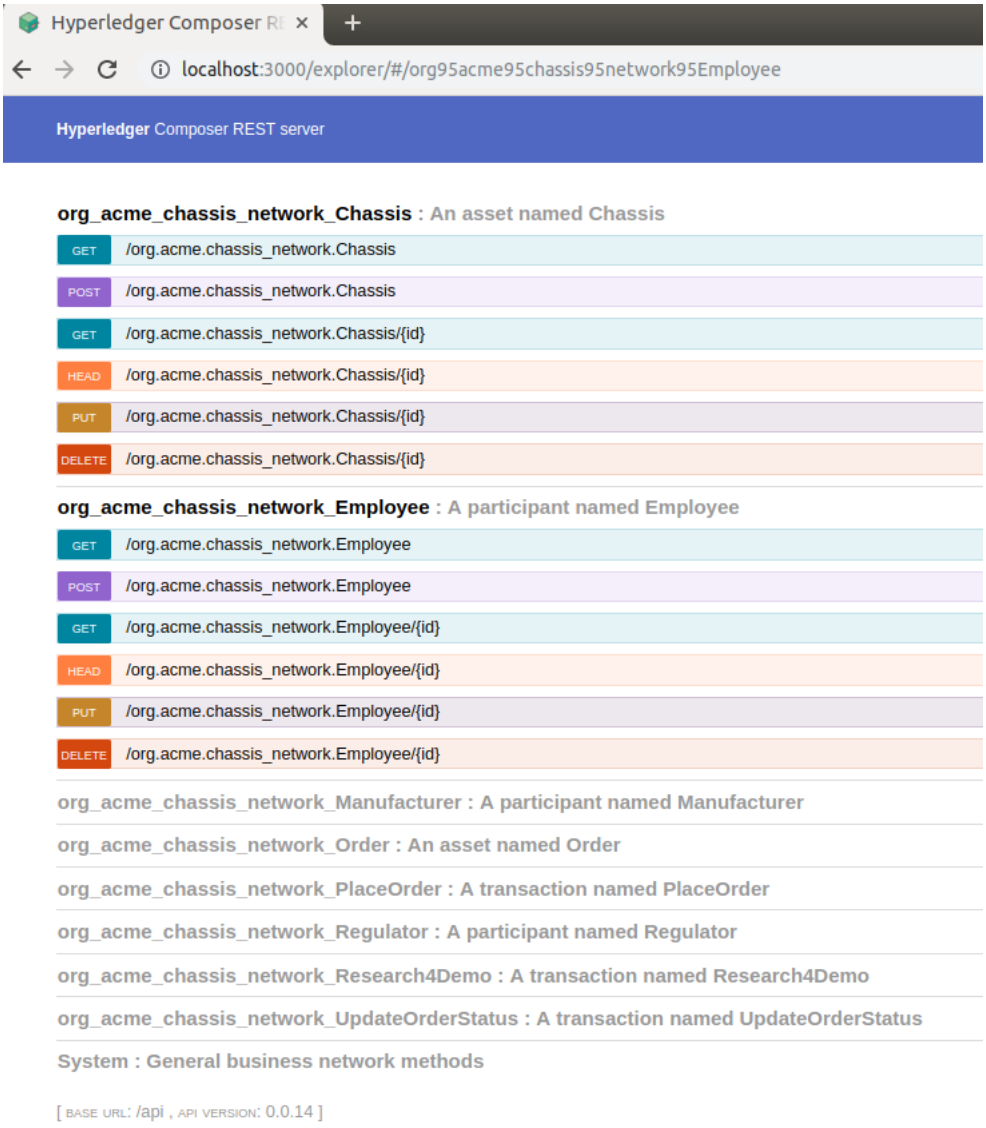

Figure 4.14: Hardware Supply-Chain REST API (inspired by [3])

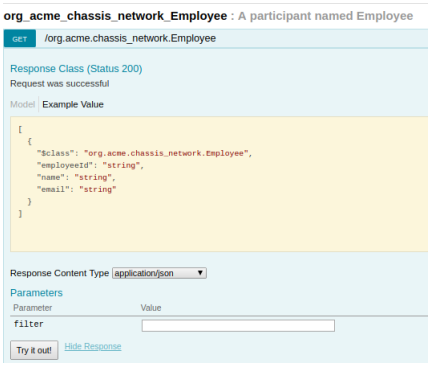

Figure 4.15: Hardware Supply-Chain GET API for *Employee* (inspired by [3])

Chapter 5

Hardware Supply-Chain Network

The ***Hardware Supply-Chain*** project is a supply business network that solves the transparency problem and enables access control restrictions. This system is using Hyperledger Composer framework for modeling the business network: participants, transactions and assets and connects to the Hyperledger Fabric via its API to add transactions in the Blockchain database and to manage participants' access accordingly with their roles in the network. Therefore, the Hyperledger Fabric is suitable for solving the hardware devices supply chain transparency and bottleneck issues and to restrict the members' access in accordance with their position in the company.

5.1 ISG CapEx procedures

ISG [2] is one of the newly acquisitions of Keysight Technologies that focuses on developing software applications for testing the network performance, getting network visibility by monitoring the traffic and finding security threats before compromising any data. Therefore employees need specialized hardware devices such as routers, gateways, switches to more powerful machines as packet brokers, ESXI servers, clusters, etc. The ISG division offers deep expertise in network testing, security and visibility which requires investing in powerful hardware equipment. In each quarter of the financial year the ISG division has a CapEx [3] budget to invest in hardware so each team can request new hardware devices. Every manager has the responsibility of identifying the hardware requirements by consulting his/her team and assuring that the total cost doesn't exceed the budget in which case the CapEx list should be prioritized.

When the CapEx requirements list is completed, the manager can proceed to order them. For ordering the equipment he/she has to send emails for approval and after the order is accepted by a superior, the manager can added it in an internal platform.

The RMA [4] procedure can be performed by the team manager as soon as he/she is informed about the hardware malfunction. Most hardware devices have an warranty period where they can be replaced or repaired free of charges by the manufacturing divisions. Before sending it to RMA the employee has to get approval from his/her manager who also needs to ask for approval from the superior.

Taking all these into consideration, the CapEx can become quite exhausting and if something happens with the hardware it is hard to find the root cause of the issue. In order to solve the

[2] Ixia Solutions Group
[3] Capital Expense
[4] Return Merchandise Authorization - return a product under warranty to receive a refund, repair, etc.

issue of transparency when tracking a device and to reduce the time of getting all the required approvals we propose a Blockchain based solution that allows everyone to check the status of orders and hardware devices. More than that we want a solution that offers immutability so nobody can modify the history of a device which is one of the core features of the Blockchain protocol. The idea is to build a permissioned Blockchain application that solves the transparency and immutability problems. In addition, it needs to be able to offer support for managing the network access control. For reasons of competitiveness a business has to keep its internal procedures private and it is important to use a solution that enables only authorized members to get access to the system data. At the end of the evaluation stage, we found two frameworks developed by Hyperledger community that meet these business requirements so we decided to use them to build our hardware supply-chain system that can replace the complicated CapEx procedures.

5.2 Hardware Supply-Chain Use Cases

The hardware supply-chain offers support for two main uses cases. The first use case solves the new hardware ordering procedure and the second use case solves the situation when a device is due to service (RMA).

In order to test the Hardware Supply-Chain system we used the user interface defined by Hyperledger Composer called *Playground* [4] and we created instances for all types of participants and three types of network card for each of them.

In Figure 5.1 we can see the three network cards used by the three types of participants: *employee@keysight*, *hardware@keysight* and *regulator@keysight.com*.

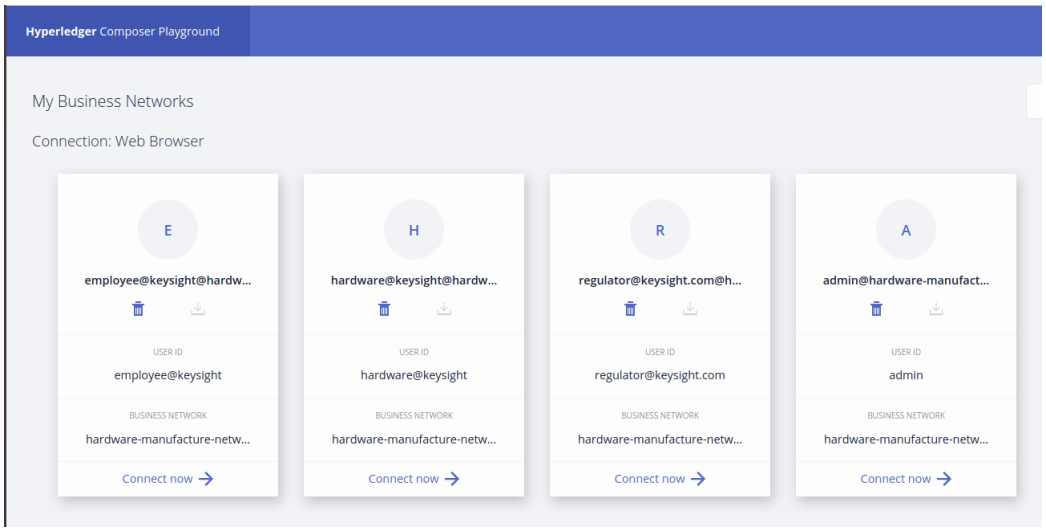

Figure 5.1: Hardware Supply-Chain: Playground (UI offered by [4])

The first use case flow is triggered by an employee by creating an asset of type *Order* via a transaction of type *PlaceOrder* (see Figure 5.2).

Each order has assigned a manufacturer that is in charge and authorized to create the device. The manufacturer analyzes the order and checks if the requested device is available or need to be scheduled for manufacturing. If no device is available the manufacturer submits a transaction of type *UpdateOrderStatus* and update the order status in *SCHEDULED_FOR_MANUFACTURE* (see Figure 5.3).

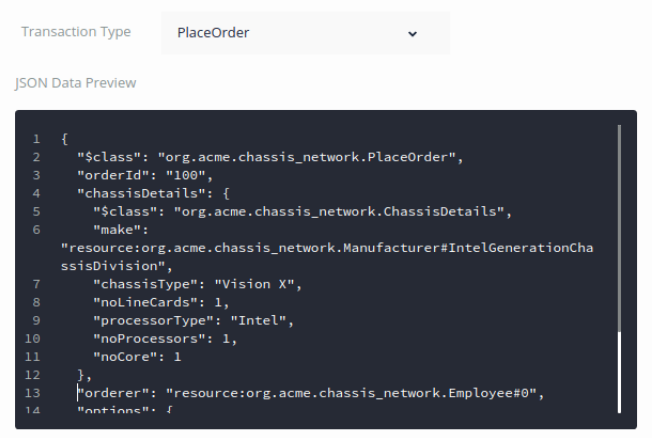

Figure 5.2: Hardware Supply-Chain: Example of *PlaceOrder* Transaction (UI offered by [4])

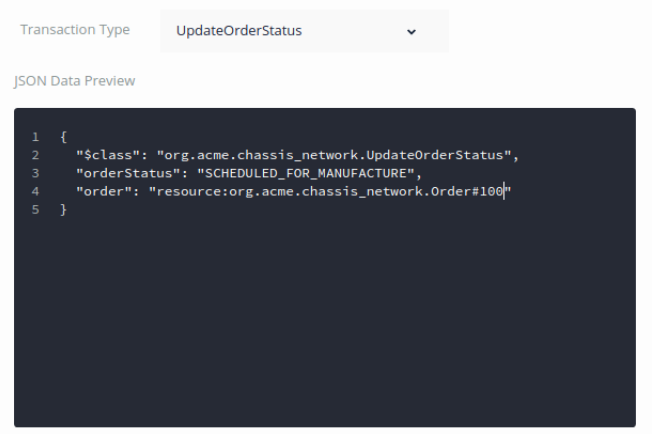

Figure 5.3: Hardware Supply-Chain: Order Status *SCHEDULED_FOR_MANUFACTURE* (UI offered by [4])

When he finishes creating the hardware ordered, he/she sets a serial number for the device by submitting a transaction of type *UpdateOrderStatus* with the status *SERIAL_NUMBER_ASSIGNED* and a unique value for the field *serialNumber* (see Figure 5.4).

In this moment, a new chassis is added in the list of assets stored in the Blockchain that can be easily identified via its serial number (see Figure 5.5).

In the end, the manufacturer has to send the device to the employee that ordered it. This step is completed by submitting a transaction of type *UpdateOrderStatus* and change the order status to *DELIVERED* (see Figure 5.6). The manufacturer is also responsible to correctly inform the delivery company all the legally required details about the packet and to set the recipient of the packet to be the direct manager of the employee that ordered the device. This use case flow ends when the designated manager receives the hardware and assigns it to the employee that ordered it via a transaction of type *UpdateOrderStatus* (see Figure 5.7). So the order changes its status in *OWNER_ASSIGNED* and the chassis has a new property: *owner* (see Figure 5.8).

Another valid scenario is the use case when the device is not in accordance with the order specifications. The owner of the device has the responsibility to check that the technical specifications of the device meet the order specifications and test the functionality of the new hardware. If something is not as expected the employee has to inform his manager. The manager has the

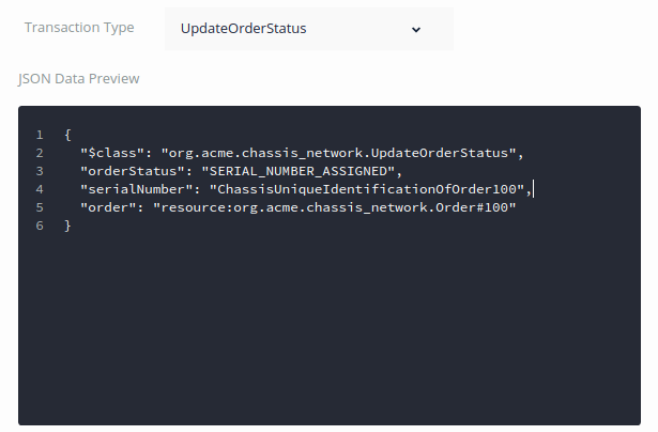

Figure 5.4: Hardware Supply-Chain: Order Status in *SERIAL_NUMBER_ASSIGNED* (UI offered by [4])


```
{
  "$class": "org.acme.chassis_network.Chassis",
  "serialNumber": "ChassisUniqueIdentificationOfOrder100",
  "chassisDetails": {
    "$class": "org.acme.chassis_network.ChassisDetails",
    "make": "resource:org.acme.chassis_network.Manufacturer#IntelGenerationChassisDivision",
    "chassisType": "Vision X",
    "noLineCards": 1,
    "processorType": "Intel",
    "noProcessors": 1,
    "noCore": 1
  },
  "chassisStatus": "OFF_THE_ROAD"
}
```


Figure 5.5: Hardware Supply-Chain: The manufactured chassis (UI offered by [4])

role of *Regulator* so it is his duty to return the device to the manufacture. If the manager approves the return of the device, he has to submit a transaction of type *UpdateOrderStatus* to change the status in *NOT_IN_ACCORDANCE*. In order to illustrate this use case we created a new order with *orderId* 1000 that has the same specifications as order with *orderId* 100 and added a new transaction of update status for the use case when an employee receives a device that doesn't meet the order' specifications. For more details see Figure 5.9. The *Manufacturer* that had registered the device serial number has to manage the use case when a device is return to manufacture. He can also check in the business network the status of the returned devices (see Figure 5.10) and the order status (see Figure 5.11).

All the submitted transactions are registered and stored in the Blockchain ledger so nobody can corrupt that data. Having an immutable ledger is one of the powerful advantages of the Blockchain usage. The Hyperledger Composer Playground offers also the possibility to read the information stored in the ledger. The first transactions stored in the ledger are issued by the Hyperledger Composer and Fabric frameworks as we can see in Figure 5.12. In our scenario we issued five transactions: one of type *PlaceOrder* and four of type *UpdateOrderStatus* (status updates: *SCHEDULED_TO_MANUFACTURE*, *SERIAL_NUMBER_ASSIGNED*, *DELIVERED* and *OWNER_ASSIGNED*). In Figure 5.13 we can see the Blockchain ledger with the issued transactions.

The second use case is similar with the one described above with the mention that the stage of delivery may be missing. In this situation the status of a chassis is set to *SCRAPPED*.

The user interface was provided by Hyperledger Composer Playground but the business model archive can be integrated with other web frameworks via the *composer-rest-server* module. The

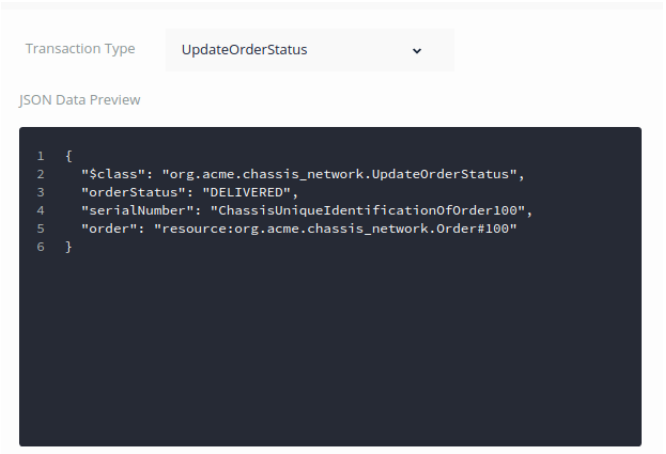

Figure 5.6: Hardware Supply-Chain: Order Status *DELIVERED* (UI offered by [4])

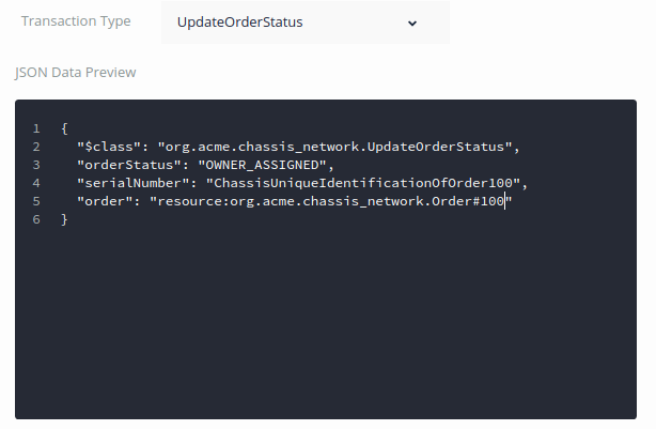

Figure 5.7: Hardware Supply-Chain: Order Status *OWNER_ASSIGNED* (UI offered by [4])

business network can be deployed on this server and we can connect from any web client to the REST API it exposes. More information on how we can use the Composer REST Server are presented at the end of section 4.2.

```
{
  "$class": "org.acme.chassis_network.Chassis",
  "serialNumber": "ChassisUniqueIdentificationOfOrder100",
  "chassisDetails": {
    "$class": "org.acme.chassis_network.ChassisDetails",
    "make": "resource:org.acme.chassis_network.Manufacturer#IntelGenerationChassisDivision",
    "chassisType": "Vision X",
    "noLineCards": 1,
    "processorType": "Intel",
    "noProcessors": 1,
    "noCore": 1
  },
  "chassisStatus": "ACTIVE",
  "owner": "resource:org.acme.chassis_network.Employee#0"
}
```


Figure 5.8: Hardware Supply-Chain: Chassis' Owner (UI offered by [4])

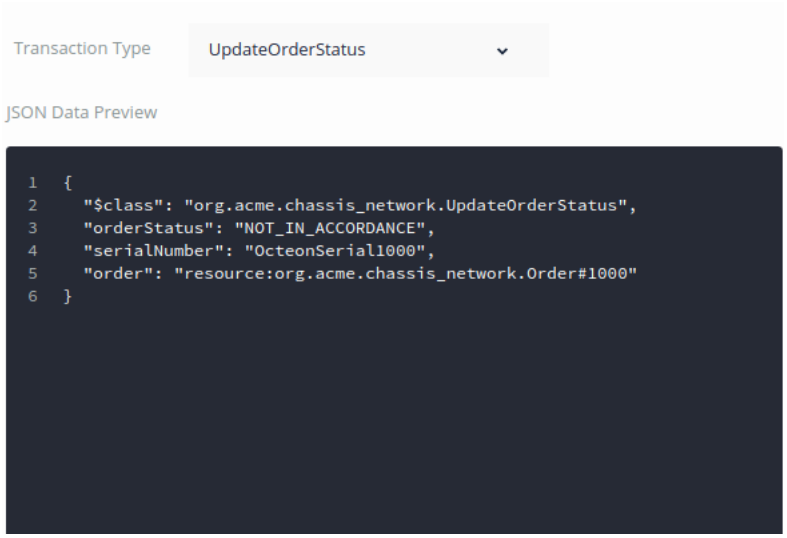

Figure 5.9: Hardware Supply-Chain: Order Status *NOT_IN_ACCORDANCE* (UI offered by [4])


```
{
  "$class": "org.acme.chassis_network.Chassis",
  "serialNumber": "OcteonSerial1000",
  "chassisDetails": {
    "$class": "org.acme.chassis_network.ChassisDetails",
    "make": "resource:org.acme.chassis_network.Manufacturer#IntelGenerationChassisDivision",
    "chassisType": "Vision X",
    "noLineCards": 1,
    "processorType": "Intel",
    "noProcessors": 1,
    "noCore": 1
  },
  "chassisStatus": "RETURN"
}
```


Figure 5.10: Hardware Supply-Chain: Chassis status *RETURN* (UI offered by [4])

```
{
  "$class": "org.acme.chassis_network.Order",
  "orderId": "1000",
  "chassisDetails": {
    "$class": "org.acme.chassis_network.ChassisDetails",
    "make": "resource:org.acme.chassis_network.Manufacturer#IntelGenerationChassisDivision",
    "chassisType": "Vision X",
    "noLineCards": 1,
    "processorType": "Intel",
    "noProcessors": 1,
    "noCore": 1
  },
  "orderStatus": "NOT_IN_ACCORDANCE",
  "options": {
    "$class": "org.acme.chassis_network.Options",
    "comment": "atip team",
    "returnComment": ""
  },
  "orderer": "resource:org.acme.chassis_network.Employee#0"
}
```


Figure 5.11: Hardware Supply-Chain: Order status *NOT_IN_ACCORDANCE* (UI offered by [4])

Entry Type	Participant	
ActivateCurrentIdentity	none	view record
StartBusinessNetwork	none	view record
IssueIdentity	none	view record
AddParticipant	none	view record

Figure 5.12: First Blockchain Ledger Transactions (UI offered by [4])

Entry Type	Participant	
UpdateOrderStatus	ChassisManufactureRegulator (Regulator)	view record
UpdateOrderStatus	IntelGenerationChassisDivision (Manufacturer)	view record
UpdateOrderStatus	IntelGenerationChassisDivision (Manufacturer)	view record
UpdateOrderStatus	IntelGenerationChassisDivision (Manufacturer)	view record
PlaceOrder	0 (Employee)	view record

Figure 5.13: Hardware Supply-Chain: Blockchain Ledger (UI offered by [4])

Chapter 6

Conclusion

This thesis summarizes our contribution to the Blockchain for business research field and proposes future directions of research. We claim to accomplish our objectives by presenting two possible classifications of Blockchain for business areas and we had identified the most popular Blockchain based projects that are specialized in meeting business requirements. In chapter 2 we describe two possible ways of classifying the Blockchain based systems: the first approach is reflecting the current state of the Blockchain for business field while the second approach provides insights in the future of this technology.

In order to validate the assumption that Blockchain technology can be easily adopted by any business we created a proof-of-concept to solve the issues identified in tha CapEx procedures conducted by the Network Visibility Solutions department of Ixia Solutions Group using two business oriented Blockchain based projects: Hyperledger Composer and Fabric. The two frameworks developed by the Hyperledger organization are programmer-friendly so anyone with basic programming skills can manage to develop a Blockchain based application in a few months. In the end, we improved our technical skills with new technologies and we broaden our mind by understanding how economical and social problems can be reduced once Blockchain becomes widely used.

There are still interesting Blockchain for business unexplored directions such as creating a business analysis for every identified area about the state of the current applications and what are the challenges they are facing alongside our solutions to help them grow their business. Also our proof-of-concept system can be improved by adding a basic web client to complete the Hardware Supply-Chain user experience and extending the network to external organizations in order to obtain a full traceability of the CapEx orders.

Appendix A

Hardware Supply-Chain REST Server

A.1 start-rest-server.sh

```
#!/bin/bash

# creates the bna with all the business network files
composer archive create --sourceType dir --sourceName hardware-manufacture-network -a hardware-manufacture-network-1.0.0.bna

# generates the PeerAdmin card
# fabric should be up and running before deploying the bna archive
composer network install --archiveFile hardware-manufacture-network-1.0.0.bna --card PeerAdmin@hlfv1

# generates the network admin card: admin@hardware-manufacture-network.card
# starts the Loopback interface on the machine
composer network start --networkName hardware-manufacture-network --networkVersion 1.0.0 --card PeerAdmin@hlfv1 --networkAdmin admin --networkAdminEnrollSecret adminpw

# restart composer-rest-server after creating the Loopback interface
# test the generated REST API directly from the explorer web interface
composer-rest-server -c admin@hardware-manufacture-network -n always -u true -w true
```

Listing A.1: Start Composer REST Server for Hardware Supply-Chain BNA

Bibliography

[1] Hyperledger Composer. Introduction. Published on hyperledger.github.io, https://hyperledger.github.io/composer/v0.19/introduction/introduction.html. Accessed Mar. 2019.

[2] Sample hyperledger composer business network definitions. Published on github.com, https://github.com/hyperledger/composer-sample-networks.git, 2017. Accessed Mar. 2019.

[3] Hyperledger Composer. Hyperledger composer rest server. Published on hyperledger.github.io, https://hyperledger.github.io/composer/v0.19/reference/rest-server. Accessed May 2019.

[4] Hyperledger Composer. Composer playground. Published on github.com, https://hyperledger.github.io/composer/latest/index.html. Accessed Mar. 2019.

[5] Alex Tapscott Don Tapscott. Blockchain revolution: How the technology behind bitcoin and other cryptocurrencies is changing the world, 2016.

[6] Nakamoto Satoshi. Bitcoin: A peer-to-peer electronic cash system, 2008.

[7] Jeroen Rijnbout. Byzantine consensus through bitcoin's proof-of-work. Published on MCA, https://pdfs.semanticscholar.org/0d82/a314dd96265ac8542a9ca80e5c7db09594d2.pdf, 2017. Accessed Apr. 2019.

[8] Ameer Rosic. What is blockchain technology? a step-by-step guide for beginners. Published on blockgeeks.com, https://blockgeeks.com/guides/what-is-blockchain-technology/, 2019. Accessed Apr. 2019.

[9] Organisation for economic co-operation and development. Published on Oecd.org, https://www.oecd.org/about/, 2019. Accessed Apr. 2019.

[10] Vita Bortnikov Christian Cachin Konstantinos Christidis Angelo De Caro David Enyeart Christopher Ferris Gennady Laventman Yacov Manevich Srinivasan Muralidharan Chet Murthy Binh Nguyen Manish Sethi Gari Singh Keith Smith Alessandro Sorniotti Chrysoula Stathakopoulou Marko Vukolić Sharon Weed Cocco Jason Yellick Elli Androulaki, Artem Barger. Hyperledger fabric: A distributed operating system for permissioned blockchains. Appears in proceedings of EuroSys 2018 conference, https://arxiv.org/abs/1801.10228v2. Accessed Mar. 2019.

[11] Hyperledger Fabric Community. Hyperledger fabric documentation. Build with Sphinx and Read the Docs, https://hyperledger-fabric.readthedocs.io/en/latest/whatis.html#, 2019. Accessed Mar. 2019.

[12] Hyperledger Composer Community. Hyperledger composer documentation. Published on Hyperldeger Github, https://hyperledger.github.io/composer/latest/introduction/introduction.html, 2019. Accessed Mar. 2019.