Analysis Report

Windesheim

Spark! Living Lab Conditioned Goods

Version 1.0

Authors:

Name	**Student number**	**Email**
Clément Perucca	s1177124	s1177124@student.windesheim.nl
Florian Paul	s1177122	s1177122@student.windesheim.nl
Thibaut Ribe	s1177132	s1177132@student.windesheim.nl
Nick van de Poel	s1139131	s1139131@student.windesheim.nl
Mischa Heimenberg	s1111178	s1111178@student.windesheim.nl

Version history

Version	When	Who	What
0.1.0	2021-10-08	Nick van der Poel, Mischa Heimenberg, Clement Perucca	Finished initial draft of analysis report.
0.2.0	2021-10-20	Nick van der Poel	Wrote introduction, blockers & stoppers, opportunities
0.2.1	2021-10-20	Mischa Heimenberg	Wrote The previous iteration chapter.
0.2.2	2021-10-22	Nick van der Poel	Renamed chapter 2 to challenges and processed feedback for it, unfinished modifications to the chapter opportunities
0.2.3	2021-10-26	Nick van der Poel	Added additional information to the introduction and converted sources to APA.
0.2.4	2021-10-27	Nick van der Poel	Added additional information to the introduction and small modifications to “the previous project iteration”
0.2.5	2021-10-27	Mischa Heimenberg	Added additional information on the previous project iteration and challenges.
1.0	2021-10-31	All Authors	Final Version

Distribution management

Version	When	Who
1.0	2022-01-14	Windesheim

Table of contents

1. Introduction

We are a group of students from different nationalities and backgrounds who have come together to work on behalf of the Windesheim Zwolle on a project for the external organisation Spark! Living Lab.

We are all attending the security engineering minor at Windesheim Zwolle through which we have gotten the opportunity to work with Spark! Living Lab on their “Conditioned goods” project. We are the third iteration of developers working on the blockchain project. We hope to deliver a blockchain application that will be usable by Spark’s client LambWeston.

We define data-assurance as the process(es) we use to increase the trustability and integrity of the data-collected by our application[1][2].

Spark! Living Lab is a research company focused on applying IoT and blockchain to the (cold) supply chain industry. Spark accepts projects from various companies depending on a (cold) supply chain. Spark’s goal is to try and figure out how their clients can benefit from IoT and/or blockchain. Currently a lot of companies are experimenting with applying blockchain and/or IoT and are looking for a way to adopt these technologies, that’s where Spark steps in.

LambWeston-Meijer (LWM) is one of those clients looking to apply IoT and blockchain to their cold chain. LWM is a large manufacturer of french fries and other processed potato products. They distribute their goods all over Europe using coldchain. In this kind of chain, the produced goods are kept in controlled environments targeting cold preservation of goods. From the moment they are being processed in the factory until the moment they are stored in the customers’ fridge, the temperature is to be kept within pre-agreed boundaries. Something that is monitored manually at the moment.

We believe security is a big aspect of the future of IT and want to explore the subject and familiarise ourselves with it. Through the “Conditioned goods” project we hope to be able to be part of a project which brings advancements to the future of blockchain trust and security while being given the opportunity to explore our interests, develop our professional skills and improve our knowledge about security and blockchain.

This document has been created to evaluate the current state of Spark! Living Lab’s Conditioned Goods blockchain network. It aims to clarify and inform the reader about some of the opportunities that Spark! Living Lab can utilize to maximize data-assurance. Answering the question: **“Which changes can be made to LambWeston’s case to maximise data-assurance inside of their Hyperledger Fabric network?”.**

In this document we will first touch upon the state of the project that the previous group left behind. After that we will discuss some of the challenges that we experienced along the way. Then we will proceed to talk about the opportunities of data-assurance within Hyperledger Fabric and specifically within smart contracts. Finally we will wrap this document up with a set of relevant requirements for this project. We have gathered these requirements through various methods such as studying the previous iterations documentation, interviews and brainstorm sessions.

2. The previous project iteration

2.1 The situation

There have been two prior iterations of the project and each of those have delivered their own products upon which this third iteration should be built upon.

The first iteration aimed to research whether blockchain could be applied to the cold chain of LambWeston to record the different states of the cargo throughout its logistical lifetime. They concluded[5] a proof of concept should be built to determine the technical feasibility of the concept.

The second iteration of the project aimed to build a proof of concept of a blockchain system consisting of the entire cold chain life cycle of involved companies with an example chain code. Once finished design documentation, both a demonstration of the chaincode and advice to involve an expert of blockchain and Hyperledger Fabric was given. All their documents can be found inside the GitHub repository[5].

During the initial conversations with Spark we have been told their expectations and goals for the project and what they aim to achieve with the system they have been building. It was made clear to us they expected the following goals are to be achieved by us during our time working for them:

Try to connect the sensors provided by InnoTractor to the HLF-network that the previous group left behind.
Conduct a risk analysis containing events related to data integrity, security and propose solutions for every event described in the risk analysis.
Answer where access-control should be applied in this project's infrastructure, this could be within the API or the smart contracts.
Edit the smart contract so that the security and integrity of the data is ensured.

Upon receiving the previous group's product we hoped to be able to continue to build upon the work they had done. However even though the product they delivered was mostly set up according to the requirements of the consortium, it was not functional. Upon building their product we ran into errors preventing us from using a working product to build upon.

We have reached out several times to members of the previous group to get an understanding of the project and the product they delivered, in those meetings we discussed the following.

In the first of three meetings we discussed the situation they found themselves in and things we should be aware of going into the project and shared some knowledge resources.

In the second meeting our goal was to get the previous iterations system working so we could use it to develop our own goals. The meeting was aimed to resolve technical issues and errors of the existing system.

In the third meeting with a member of the previous iteration we asked how they went about testing and demonstrating their chaincode without a working system to which we had a demonstration.

2.2 Smart contract analysis

We have conducted an analysis of the previous iteration of smart contracts to understand how they work and what types of improvements could be made to the environment.

2.2.1 Definition and context

The smart contracts are functions written in typescript. The smart contracts are based on the fabric-contract-api[6] which is a NodeJS library created by Hyperledger Fabric.

There are two contracts existing:

- The measurement contract who aims to verify if the measures for the temperature's sensors are valid.
- The shipment contract who handles the shipments in the ledger. You can create shipments and add sensors to them.

2.2.2 Measurement contract

In this paragraph we are going to talk about each function present in the measurement contract, what their uses are and then review the main problems.

Functions:

The `Measurement` is an object containing three fields; `sensorID`, `value` and `timestamp`.

getMesurement: This function is used to get a measurement from a shipment.

getHistory: This function gives the history of the measurements.

addMeasurement: This function is used to add a measurement to a shipment.

validateSLA: This function is used to validate the SLA (service level agreement). It checks that the `value` field of a `Measurement` instance is between two constants predefined (minimal and maximal temperature).

Review:

The main issue is clearly that anyone can create a measurement that implies that the data cannot be trusted. The second main issue is that the only verification in the contract is that a temperature is included between two others: can be improved for example it could verify the position or the owner of the shipment.

2.2.3 Shipment contract

In this paragraph we are going to talk about each function present in the shipment contract, what their uses are and then review the main problems.

Functions:

addShipment: This function creates a `Shipment` instance. A `Shipment` instance has the following fields: `id`, `temperature`, `sensors` and `createdAt`.

updateShipment: This function updates shipment in the ledger. It is used to set new values for a shipment. You could for example update the temperature using this function.

getShipment: This function is used to get a shipment from the state database. It is then possible to modify it before updating it back.

shipmentExist: This function checks whether a `Shipment` with the given identifier exists.

getShipments: This function executes a query to get all the shipments from the ledger and returns an object containing the list of the shipments.

registerSensor: Register sensor on a shipment: a sensor can only be attributed once to the shipment. However, it seems like you can register a sensor which is already used on another shipment.

sensorIsRegistered: Check if a sensor is registered on a shipment. It returns the ids of the sensors attributed to one shipment.

getShipmentBySensor: This function is used to get a shipment from the ledger thanks to its id.

getShipmentBySearchString: This function is used to return the first `Shipment` instance matching the search query matching the identifier.

Review:

- There is no access control. Anyone can create shipments, add sensors. No system of ownership for the shipments. Apparently, there is no verification to update the shipments.
- You can register one sensor to multiple shipments.
- A shipment doesn't have any owner. If there are multiple peers this can be a problem because only the one who is currently using the shipment should be able to add measures.
- A shipment doesn't have any state like delivered or waiting for check up. That also means that you can say you've delivered a shipment that has already been delivered.
- When the SLAs are validated the putState function is not called so nothing is written in the ledger. Therefore there is no trace of validation in the ledger.
- When the SLAs are checked they don't verify the origin of the value(from which sensors it comes from). You can give an id of a shipment but hence it's possible to have multiple sensors on the same shipment, how do you know which one you have to check?

- No use of the getState function to have some knowledge about the current state of a shipment in the blockchain. So, the code never checks if a shipment was already delivered for example.

3. Challenges

We have dealt with numerous challenges while finding our way inside of the project. Some signs as early as in the stage where we tried to establish our project plan. In this chapter we will discuss the challenges we came across during the entire duration of our iteration.

3.1 Physical sensors

One of the initial goals we mentioned earlier for this project iteration was to connect a physical sensor to the Conditioned Goods HLF-network. For a month we had been waiting for InnoTractor to provide the sensors to us. Then Spark had a meeting with InnoTractor and got to hear that there were still issues with their sensors and thus unable to provide the sensors to Spark. The reasons for this:

- Their sensors were not ready for temperatures below -15°C.
- Their sensors could not collect batches of measurements when they were disconnected from the internet.
- Their sensors were not certain for battery life.
- Their sensors still weren't tested for blockchain.

Due to this unexpected news we were unable to meet the physical constraint of the initial goal to connect a sensor to the blockchain. After discussing this problem with Maxime we came to an agreement to continue using mock data for sensor readings and skipping the physical constraint. Connecting physical sensors has yet to be done and confirmed to be possible.

3.2 Unfinished HLF-network deployment

It was expected that we, in project this iteration, would pick up where the previous group left off. From our first couple of meetings with Spark and explanation of the project by Windesheim, the first logical step would have been to successfully deploy the previous group's code. It would have been the first and most direct way to start adding value to the product as it would be Spark's overarching wish to develop a proof of concept for LambWeston.

After having struggled for several weeks and meeting multiple times with members from the previous group we discovered that their deployment network was not ready yet to be deployed. Up until then we had tried to get all docker containers up and functioning as one network but the latter seemed rather difficult. We had to investigate the issues making use of all the available logs. This investigation also required us to invest time in learning how for example Apache Kafka and Zookeeper worked (and their dockerized version). Fixing the deployability of their network would have required us to invest more time than we could afford due to the required deliverables of Windesheim.

With limited results we decided with Maxime that it would be more efficient to start working on the network itself rather than deploying it so that we could still focus for the remainder of time on the data-assurance aspect of this project. Finding an efficient way to deploy the production network from the previous iteration has yet to be found and is likely something for a separate iteration to work on. Quite possibly in combination with the task of connecting sensors to the environment.

3.3 Different stakeholder expectations

When we started out with analysing the current state of the Conditioned Goods project we started measuring the different expectations that would add value to the business of all related stakeholders. Quickly we discovered that Windesheim had a different understanding of this project iteration than Spark itself and in addition also Maxime seemed to have a bit different understanding from this project iteration than our project coach, Chris. It was certainly an issue to get a consistent understanding of this project iteration across the board. This resulted in more work managing the project than expected.

Windesheim expected us to learn something related to security engineering and prove what we learnt by showing proof of the following competencies: analyse, advise, design, realise and manage & control. In the end we managed Windesheim's expectations by making a switch to dedicate ourselves to writing the following documents as proof: analysis report, advisory report, design document, RTM and a proof of concept.

Spark expected us to succeed on their initial goals adding value to the existing proof of concept. In comparison to Windesheim this would have been limited to the realisation part and providing Spark! with a risk analysis. They were of the opinion that most of the documentation that Windesheim required for our assessment to be unnecessary.

4. Opportunities

While working on the project we have encountered numerous opportunities. Some of these opportunities have presented themselves while we ran into issues, others we have discovered while analysing the existing setup or researching developments related to blockchain. In this chapter we will discuss these opportunities

- After realising we were unable to get the production environment running we started researching the opportunities. We came to the conclusion we should build a single node system allowing us to continue development in a limited scope. However this path of development brought some advantages.

  Focussing on a single node system to develop our project means there are less vectors of interruption and complications as the system we develop on is less complex. This allows us to focus on the development without having to spend as much time troubleshooting possible issues and complications.

- Using Raft as our consensus algorithm. The problem with Kafka is that you will need one Kafka cluster for each organisation. However with Raft, each organisation can have its own ordering nodes, participating in the ordering service, which leads to a more decentralised system.

- As we implemented the notion of ownership, the transfer of ownership was necessary to implement in the system. In order to match with our vision of the lifecycle, we create a transfer by two-ways agreement. In fact, the transfer of ownership to be agreed between the actual owner and the future owner. Consequently, we created a system where the actual owner needs to send a request to transfer the ownership of the transport and then the future owner accepts that transfer. During all this process, we also check that the SLA is verified.

- Configtx.yaml (HLF channel configuration)
  We found that the configtx file is managing the channel configuration and consequently the classification of each organisation. We configured it in our code but not completely because of a lack of time. Thanks to that, it is possible to manage access control without implementing it in smart contracts which make it easier to deploy and to manage. This implies that we do not have to redeploy smart contracts every time that we want to add an organisation to the blockchain.

- Although cryptogen is often provided to be a simpler alternative to fabric-CA, it is recommended that it should only be used in development. Cryptogen only generates self-signed certificates and that’s why the certificate cannot be verified by a 3rd-party. Fabric-CA also requires new users to be registered and enrolled by the CA-admin of that organisation.

5. Requirements

Below we have listed a number of requirements which are relevant to the project. These requirements will be continued on in the design documentation. We have gathered these requirements through various methods such as studying the previous iterations documentation, interviews and brainstorm sessions.

We have given these requirements a number of attributes. These attributes consist of the requirement number as listed in the requirement matrix, a description of what the requirement specifically entails, the type of requirement whether it's functional, qualitative or a principle, the priority of the requirement within the project and finally the source where we gathered the requirement from.

N.	Description	Type	Priority	Source
1	The sensor must include a timestamp in the measurements	Functional	Must have	Previous iteration
2	Sensors must be able to communicate with the blockchain	Functional	Must have	Previous iteration
3	Unauthorised users cannot read data from the blockchain	Functional	Must have	Spark!
4	Unauthorised users cannot update the blockchain	Functional	Must have	Spark!
5	The chaincode must allow for SLA compliance in the blockchain	Functional	Must have	Previous iteration
6	The majority of the organisations need to approve new chaincode	Principal	Must have	Project team
7	The Blockchain infrastructure should be designed for maximum organisational independence	Principal	Should have	Project team
8	The blockchain should be decentralised	Quality	Should have	Project team
9	The blockchain verifies the sensor measurements	Functional	Must have	Previous iteration
10	Only verified sensors can be applied to shipments	Functional	Must have	Project team
11	There cannot be shipments	Quality	Must have	Project team

	and sensors with duplicate identifiers in the system			
12	There is a clear state indication that the shipment has been completed	Functional	Must have	Project team
13	Only measurements from verified sensors are inserted into the blockchain	Quality	Must have	Project team
14	The smart contracts must allow for transfer of ownership of shipments	Functional	Must have	Project team
15	The API must be able to insert data into the blockchain	Functional	Must have	Previous iteration
16	The API must be able to query data from the blockchain	Functional	Must have	Previous iteration
17	The API is connected to the blockchain	Functional	Must have	Previous iteration
18	Data can be read by authorised organisations	Functional	Must have	Project team

6. References

[1] Bloomberg, LP. (2021, July 26). How does data assurance increase confidence in data? – The ODI. Retrieved September 24, 2021, from https://theodi.org/article/how-does-data-assurance-increase-confidence-in-data/

[2] Tatum, M. (n.d.). What is Data Quality Assurance? EasyTechJunkie. Retrieved September 24, 2021, from https://www.easytechjunkie.com/what-is-data-quality-assurance.htm

[3] Deloitte. (n.d.). Using blockchain to drive supply chain transparency. Deloitte United States. Retrieved September 30, 2021, from https://www2.deloitte.com/us/en/pages/operations/articles/blockchain-supply-chain-innovation.html

[4] Vadgama, N., & Tasca, P. (2021, March 23). An Analysis of Blockchain Adoption in Supply Chains Between 2010 and 2020. Frontiers. Retrieved September 29, 2021, from https://www.frontiersin.org/articles/10.3389/fbloc.2021.610476/full

[5] Spark! Living Lab. (n.d.). conditioned-goods-use-case/reports at dev · Hogeschool-Windesheim/conditioned-goods-use-case. GitHub. Retrieved September 30, 2021, from https://github.com/Hogeschool-Windesheim/conditioned-goods-use-case/tree/dev/reports

[6] hyperledger. (2021, August 11). npm: fabric-contract-api. Npm. Retrieved September 28, 2021, from https://www.npmjs.com/package/fabric-contract-api

[7] Tam, K. C. (2021, December 12). PutState and GetState: The API in Chaincode Dealing with the State in the Ledger (Part 1). Medium. Retrieved October 31, 2021, from https://kctheservant.medium.com/putstate-and-getstate-the-api-in-chaincode-dealing-with-the-state-in-the-ledger-part-1-2008d6a86f98