UNIVERSITY OF TWENTE.

Master thesis

University of Twente

Department of Electrical Engineering Mathematics and Computer Science

Improving food safety by designing a decentralised traceability architecture in line with stakeholder concerns.

Author:
Rob van Dijk

First supervisor:
dr. ir Erwin Folmer

Second supervisor:
dr. ir. Marten van Sinderen

Formal supervisor:
dr. Christopher Brewster

Daily company supervisor:
ir. Paul Brandt

30th October 2020

Abstract

In the recent past, society was alarmed by food safety incidents multiple times. These food safety incidents hurt both consumer safety and the value creation of supply chains. One of the underlying problems that contribute to food safety incidents is the lack of traceability in the agri-food supply chain. The purpose of this thesis is to address the lack of traceability, to increase food safety. Concurrent traceability systems do not address stakeholder concerns enough. Moreover, agri-food supply chain actors still perceive barriers in the adoption of traceability systems. In this thesis, an architecture is designed to address these concerns and barriers. This architecture captures traceability data in a decentralised manner. Each supply chain actor captures their traceability data in a pedigree. The pedigree describes the product and actions done to create the product. Actors share the identifiers of the pedigrees to create linked pedigrees. These decentralised linked pedigrees contain the traceability data needed. The architecture uses the Solid ecosystem to realise the exchange and storage of data. Fine-grained access control was realised by emulating Ontology based access control (OBAC). OBAC presents a novel and efficient method to define access control policies based on the structure of the data. The architecture was validated by building a proof of concept, that was exposed to a scenario of the horsemeat scandal. The architecture is shown capable of creating traceability as well as handle uncooperative actors in the case of a food security incident. Furthermore, this research provides a use case for the Solid ecosystem outside of the initial social media context. Next to that, the architecture could be used in other domains that require traceability.

Contents

1 Introduction

In recent years, European residents' confidence in the quality of their food is lowered and they demand higher quality assurance (Bánáti, 2014). This became visible after the food scandals in the recent past, such as the infamous horsemeat scandal, melamine in milk, and fipronil in eggs. Four weeks after the horsemeat scandal, a decline of 43% in frozen burgers and 13% in ready-made meals sales was observed (Kantar Worldpanel, 2013). According to the Grocery Manufacturers Association (GMA), the melamine adulterated milk had a total cost of ten billion dollars. The GMA further estimates that "The cost of one adulteration incident averages between two and fifteen per cent of yearly revenues depending on company size" (GMA & Kearney, 2010).

A review of literature and media in 2013 found 137 distinct food fraud cases since 1980 (Everstine, Spink, & Kennedy, 2013). In the seafood industry, multiple studies found percentages of 25, 26, 37 and 33 of wrongly labelled, more expensive species. In 2008, 22 Chinese food companies sold dairy products with melamine. 300.000 children became ill and 6 had died. Another tragic case happened in 1981 in Spain when consumers bought denatured oil, intended for industrial use, as olive oil. This resulted in almost 20.000 cases of sickness and more than 300 deaths. These are some of the extreme cases (Everstine et al., 2013).

There are four root causes for food safety incidents are; long and complicated supply chains, a lack of resource dedication, power asymmetry, and a lack of traceability along the supply chain. The root causes are explained in more detail below.

- **Long and complicated supply chains**

  Long and complicated supply chains create fraud opportunities (Everstine et al., 2013). To reduce the cost of beef, traders searched for cheaper sources in other parts of Europe. Ultimately this led to complex supply chain networks to gain cheaper resources in other parts of Europe (Czinkota, Kaufmann, & Basile, 2014). The expansion of the supply chain led to an increased amount of actors. Each new actor creates a new place where contaminants can be introduced. Next to that, the complexity of the supply chain increases the chances for criminals to stay hidden. In the horsemeat scandal, the search for cheap beef eventually led to the introduction of horse meat into the supply chain.

- **Lack of resource dedication**

  "Sustainable quality cannot be achieved without dedicating sufficient resources by all stakeholders and there is a limit to budget cutting on a macro level" (Czinkota et al., 2014). Product quality will eventually drop when budgets are cut to produce as cheaply as possible. For example, cheaper products can be made when fewer quality tests are conducted, increasing the likelihood of an undetected incident.

- **Power asymmetry**

  Power asymmetry in a single supplier - multiple buyer relationships contributes to food adulteration (Nnamdi O. Madichie & Yamoah, 2017). A single supplier - multiple buyer relationship creates an environment with an over-reliance on a single supplier. This over-reliance increases risks for organisations when something happens with this supplier. For example, in the horsemeat scandal or banana production after Hurricane Mitch in 1998 (Hittle & Leonard, 2011). Actors could no longer keep up with demand because crops were ruined, lowering production capacity. In times without food safety incidents, investing in capacity flexibility reduces efficiency. However, history shows that reoccurring supply chain crises make it worthwhile to invest in capacity flexibility (Nnamdi O. Madichie & Yamoah, 2017). The common success and failure characteristics to survive a supply chain crisis are quantitatively identified by Hittle Leonard (2011). Unfavourable characteristics include: Dependence on a single supplier and a poor supplier relationship. To survive a supply chain crisis, the best indicators are: capacity flexibility, multiple suppliers, planning for crises and effective communication along the supply chain. These studies demonstrate that the desire for operational efficiency at individual and supply chain level increase risks for the supply chain and possibly for end consumers (Hittle & Leonard, 2011; Nnamdi O. Madichie & Yamoah, 2017).

- **Lack of traceability**

  Finally, a lack of traceability in the agri-food supply chain (Bánáti, 2014; GMA & Kearney, 2010; Everstine et al., 2013; Nnamdi O. Madichie & Yamoah, 2017; Manning & Soon, 2016, 2014; van Ruth, Huisman, & Luning, 2017; Czinkota et al., 2014). The lack of traceability originates from the long and complicated supply chains. Although some methods for traceability exist, they do not enable a timely response in the case of a food safety incident. This can be observed from the response time to find the origins and destinations of contaminated products. Examples of this include the *E. Coli* contamination and the horsemeat scandal. In the case of the *E. Coli* contamination, six months after detection no origin was known (Buchholz et al., 2011). In the horsemeat scandal, five weeks after the discovery the root cause was still unknown (Felicity, 2013a). It took six months to map the supply chain. European law requires agri-food supply chain actors to keep records of procurement and sales. However, the regulation does not mention a format. At this moment, traceability is largely done on paper (Minnens, Sioen, van de Burg, Luijckx, & Verbeke, 2018). This does not enable timely reactions when food safety incidents happen because a manual search through paper records is time-consuming. The time to find causes of food safety incidents is worsened by the allowed response time of 24 hours for each supply chain partner (Welt & Blanchfield, 2012). In that case, it costs one week to gather all the required information in a supply chain of seven actors.

The disappearance of companies further increases the long times to find the origin of food safety incidents. This could be observed in the 2011 organic food crisis in Italy where, among other problems, short term companies were founded and closed to make traceability impossible (Flari et al., 2014).

Although traceability is formally in place as required by law, the long-lasting search for the origin of a food safety incident hurts both the agri-food supply chain and consumers. Due to the long search times, authorities cannot prevent consumers from ingesting harmful products. Moreover, consumers can no longer trust the food they buy is safe for consumption. Supply chain actors are stuck with potentially contaminated products that they cannot sell and have reduced overall profit.

A potential solution to increase the speed of traceability is the use of a traceability system. This solution direction is not new, as they are commercially available. E.g. TE-food, SeafoodIQ and IBM food trust. The commercially available traceability systems demonstrate that traceability systems can be created. Moreover, the willingness of actors to adopt a traceability system is not the problem either (Minnens et al., 2018). The problem is a combination of both: agri-food supply chain actors do not adopt systems that are not aligned with their concerns. Because traceability data contains sensitive data of supply chain actors, supply chain actors hesitate to adopt a system where data confidentiality is not ensured. Moreover, all the supply chain actors must use the same traceability system (or an interoperable one) to realise traceability. Multiple studies investigated the perceived barriers to adoption of a traceability system. Therefore, the design of a system that addresses the lack of traceability in a manner that conforms to the stakeholders' concerns is described in this thesis. When supply chain actors adopt traceability systems, it could realise the benefits of traceability described in section 1.2. Before the benefits of traceability are described, a definition of traceability is given.

1.1 Definitions

1.1.1 Traceability

Traceability is defined as: "The ability to access any or all information relating to that which is under consideration, throughout its entire life cycle, by means of recorded identifications." (Olsen & Borit, 2013). This definition includes upstream and downstream tracking, as well as internal (traceability inside a single company) and external (traceability between actors in the supply chain) traceability. Moreover, it means that a system can be queried for specific data. Traceability data refers to the information in the recorded identifications. This thesis focuses on external traceability, as it is assumed that organisations have measures in place for internal traceability as it is required by law to capture this (Regulation (EC) No 178/2002).

1.1.2 Food safety incident

Throughout this thesis, the term food safety incident is used multiple times. In the literature, a wide variety of terms exist for the safety of food. Therefore, all the terms and definitions in the literature i.e. economically motivated adulteration, food fraud and food defence incidents have been piled into *food safety incident* to create a word that indicates that food is unsafe for consumption regardless of the reason why. This is not in line with the definition in the literature where a food safety incident indicates unintended, harmful contamination (Manning & Soon, 2016).

1.2 Benefits of traceability

Bosona & Gebresenbet (2013) identified and summarised 6 benefits of food traceability, listed below.

1. **Increase in customer satisfaction** Traceability is expected to increase the confidence of customers in food quality. The available product information enables customers to make informed choices which increase the confidence of consumers. Additionally, it is said to reduce customer complaints and reduce social costs.

2. **Improvement in food crises management** In the case of a food safety incident, traceability can present the products affected on a detailed level which "minimises the production and distribution of unsafe or poor quality products" (Bosona & Gebresenbet, 2013). Because information is available faster, product recalls reach the affected producer faster. The affected producer can prevent further waste work on the affected product. Moreover, faster recalls of affected food products can prevent further consumption by unaware consumers. Additionally, companies can directly prove the extent of their liability in food safety incidents. Some argue that traceability reduces the chance of food safety incidents. Others argue that while traceability systems enable effective management of food safety incidents, they do not affect the chances of food safety incidents.

3. **Improvement in food supply chain management** Traceability data can increase the efficiency of logistics and reduce costs. Traceability data contain the product routing, from which more efficient routes can be distilled. Additionally, it can increase the amount of coordination between partners. The increased supply chain management creates the possibility for partners to develop their economic and technical competence.

4. **Competence development** Effective traceability systems help to stay competitive in the market by the development of competences. Because a traceability system: " i) enables to solve food safety problems; ii) provides a good-faith legal defence in product liability cases; iii) enables a company to understand well its logistics system; iv) provides promotional advantages by connecting manufacturer with consumers (Hall, 2010); and

v) enables to develop products of better quality in long run using the laboratory based test results and availability of traceability information." (Bosona & Gebresenbet, 2013)

5. **Technological and scientific contribution** Organisations that implement traceability systems help research because it generates the data required by scientific research into the cause of food safety incidents. Additionally, it provides data about products, production and the supply chain which can be used in the development of new technologies and tools used to create traceability.

6. **Contribution to agricultural sustainability** Traceability promotes transparency in sourcing and food production. The transparency helps in the implementation of sustainability initiatives. At this moment, a large proportion of seafood cannot be traced to the level of a vessel or fish farm and may be caught in a non-sustainable manner. This is caused by the aggregation of batches at auctions (Randrup et al., 2008). Traceability systems can prove seafood is caught sustainable. Additionally, traceability systems are said to promote effective packaging technologies, which reduce food losses during distribution.

1.3 Problem description

As previously mentioned, agri-food supply chain actors perceive multiple barriers to adopt a traceability system. The most prominent of these is the concern for the confidentiality of their data. However, when every actor keeps their data private, the data is not available and accessible for traceability. Therefore the goal of this thesis is to design a traceability system that conforms to the requirements of stakeholders. To do this, five research questions were created which are stated below. The first research question identifies the requirements, the second and third search for potential solutions, the fourth research question creates the artefact that addresses the goal of this thesis. The fifth research question validates if the artefact satisfies the requirements.

1. What requirements should a traceability system satisfy in the agri-food supply chain?

2. What are the possible treatments that satisfy the requirements?

3. What are the advantages and disadvantages of the available treatments?

4. What is an architecture that would satisfy the requirements for a traceability system in the agri-food supply chain?

5. How well does the architecture handle simulated food security incidents from the past?

1.4 Method and Outline

In this thesis, the design problem of: a traceability system that conforms to the requirements of stakeholders is investigated. The design cycle from design science methodology was used (Wieringa, 2014). In total three phases were done: the problem investigation, treatment design and treatment validation. In design science methodology, a treatment is a solution for a problem, hence the names treatment design and treatment validation. Each phase answers one or more research questions.

In the first phase, the problem investigation, the goal is to "identify, describe, explain and evaluate the problem to be treated" (Wieringa, 2014). This happens in the problem investigation section 2 and creates the background for the requirements of research question 1.

The next phase is the treatment design. In this phase, a design is made to solve the problems identified in the problem investigation. The treatment design provided the background to answer research questions two to five. The requirements are specified in the first step of the treatment design. The requirements are listed in section 3 and provide the answer to the first research question. After the requirement specification, the available treatments found are listed in the state of the art, section 4. These were used to answer the third research question. Finally, two state of the art systems were combined in a new treatment, specified in section 5. This section provides information to answer the fifth research question.

The third phase in design science is treatment validation. During this phase, a check is done if the designed treatment solves the defined problems in the problem investigation. The treatment validation is located in section 6 and provides the background to answer the fifth research question.

In section 7, the research questions are answered and discussed. Next to that generalisations, recommendations and suggestions for possible future work are given.

Throughout the thesis, the horsemeat scandal is used as a motivating example. This is done because it gives a concrete model from the real world to use as an example. Additionally, the lack of traceability, long and complicated supply chain, overreliance on a single partner and a lack of resource dedication are present in this example.

2 Problem investigation

In this section, the perceived barriers to adoption of traceability systems are identified. The information gathered in this section is used to answer the first research question: What are the perceived barriers by stakeholders in the agri-food supply chain to adopt a traceability system? To do this, the problem investigation of design science examines: the goals and concerns of stakeholders (section 2.2), available frameworks (2.3) and observed phenomena (2.3). This section starts with a conceptual model used to describe the orientation of stakeholders in the supply chain in relation to each other.

2.1 Conceptual model for supply chain orientation

In this thesis, communication between supply chain partners is a reoccurring subject. This model was made because organisation names, upstream and downstream could result in ambiguous sentences. In figure 1, the conceptual model for stakeholder orientation can be seen. The goal of the conceptual model is to demonstrate how different supply chain actors are positioned in the supply chain. It shows suppliers and buyers of an actor, and shows who are further upstream and downstream than the direct buyers and suppliers of an actor.

In this model, *A* is the organisation of interest and can be every actor in the supply chain. *A* can have multiple partners up and downstream. For each subsequent downstream actor, the letter *A* is increased in alphabetical order. For example, if *A* is a tomato farmer, *B* is the sauce producer. Then *C* is the lasagne producer. *A* can be the first actor in the chain (farmer) as well as the last (retailer). To indicate upstream actors, a *U* is added. In the previous example, *UB* is the seed supplier of tomato farmer *A*. A number is used to distinguish between multiple actors on the same level of the supply chain. An example: *UB1* is a minced meat supplier. *UB2* is a pasta supplier. *A* uses minced meat from *UB1* and pasta from *UB2* in his ready made-lasagne. *C* can be a supermarket who sells the lasagne produced by *A* and distributed by *B1*. For the remainder of this thesis, actors in the supply chain are indicated by italic capital letters similar to this section.

2.2 Stakeholders and their concerns

"Stakeholders are the source of goals and constraints of the project, which are in turn the source for requirements in the treatment, and so it is important to identify relevant stakeholders" (Wieringa, 2014). Therefore this section contains a list of stakeholders and their concerns to be used in subsequent sections. The list of stakeholders was derived from a list of possible stakeholders of projects (Wieringa, 2014).

- **Supply chain actors** The supply chain actors are the agri-food supply chain companies which use traceability systems to create traceability of their products. These are the farmers, food processors, distributors and supermarkets.

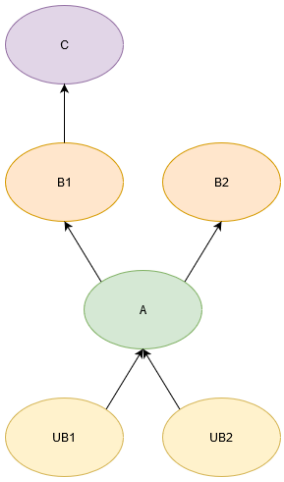


Figure 1: Model of possible interactions between supply chain actors.

The primary goal of the actors, as they are commercial entities is to create value. However, the current food scandals hurt the brand of these companies. Current traceability documentation is largely done on paper, an electronic system may reduce documentation effort and increase efficiency. Furthermore, the benefits of traceability may reduce the costs of food safety incidents. But on the other hand, there are a lot of possible concerns for supply chain actors before the adoption of a traceability system.

  - **Costs**: Will it be affordable? How will it affect the efficiency of the organisation? Will the promised benefits materialise, especially when only part of the supply chain implements traceability? Do I require new employees to use the system?
  - **Data**: Is my traceability data safe? Who has access to my data? Is the data anonymous? What can others do with the data they receive? Can I trust the data quality of other actors? Can the quality of the data be guaranteed? These concerns are discussed in further detail in section 2.3.
  - **Commercial context**: Will the use of a traceability system increase the number of audits? Will my competitors profit from my effort to implement a traceability system? Do we require another system and does it require changes to the IT landscape?

- **End users** The end users are the employees of the supply chain actors which use the traceability system. The concerns of the end users are relevant because improperly managed end users can become threat agents. For example, their concerns include:

  - Do I have enough skills to use the traceability system? The current paper-based way of working is easy to understand, I may not understand the new system.
  - Will the traceability system take more of my time than the previous (paper) documentation and increase my workload?
  - Will the traceability system threaten the existence of my job? When gathering traceability data goes faster or is automated, my job may no longer be required.

- **Software owner** This is the organisation that produces traceability systems. Its functions consist of maintaining the system, as well as granting support for new adopters and possible training. Their concern is to have a working product which is sellable. Moreover, they want to reach a critical mass of adoption of their system.

- **European food safety authorities** The food safety authorities are responsible for The quality of food in Europe. Their goal is to grant customers trust in the food they buy. A traceability system may improve audit speed. Moreover, when a food safety incident happens, traceability systems may reduce the effort required to find the root cause. Thereby better protecting the European citizens. Additionally, it may even safe work when organisations handle safety incidents before it reaches consumers. The concerns of the authorities include: can the data be trusted? What happens when data are lost?

- **Consumers** The end consumers of the products from the supply chain using a traceability system. The primary concern of consumers is: is my food safe? They will benefit from traceability systems by a reduced chance of food safety incidents reaching the consumer. Depending on the implementation it may be possible to observe where their food originated. This is particularly valuable for communities that require food containing undetectable properties, such as fair trade.

- **Current traceability systems** The traceability systems that are on the market at this moment. A new traceability system may pose a threat to their market share. Their reason for being is to create a profit of their system. Their goal is to gain adoption of their system over others.

- **ERP systems** End users of a traceability system may have ERP systems that produce traceability data that can be reused. Because some functions may overlap, the developers of the ERP system may feel threatened. Their concerns are: Will a new traceability system cost us market share? Will my ERP system become obsolete?

- **Fraudsters** Any traceability system will require data to work. Fraudsters may provide the system with faulty information which lowers the value of the system. They will want to keep their (dark) market share. Traceability

systems are negative for them, and fraudsters will, therefore, attempt to hinder the adoption of traceability systems. Because their network may be throughout the entire supply chain, they may prevent the total adoption of any traceability system.

2.3 Barriers to adoption

In this section, a list of found factors affecting software adoption decisions is given. For each factor, an explanation and the perceived barriers are given. The list of factors is based on literature about factors that drive or inhibit software adoption. This literature empirically tests factors that influence the adoption decision for software as a service, ERP systems and anti-spyware. The papers build on frameworks like TOE, VAM and theory of planned behaviour to create models for determinants. The factors below are a summary of found factors and combined into categories according to the writers' interpretation. The factors may not be entirely correct because all studies rely on different models, methods, backgrounds and definitions. But they still provide factors worth considering. Applicability of the factors to the agri-food domain is estimated based on the literature on perceived barriers to adoption.

Ease of use

The (perceived) ease of use positively affects the adoption chances (Kwon & Seo, 2013; Seethamraju, 2015; Wu, 2011; Yang, Sun, Zhang, & Wang, 2015). However, not for anti-spyware (Lee & Kozar, 2008).

This is reflected in the literature on perceived barriers. Some studies indicate that companies lack skilled employees to use the traceability system (Bosona & Gebresenbet, 2013; Duan, Miao, Wang, Fu, & Xu, 2017). Next to that, as it is allowed to store traceability data on paper, the transition to a digital system could be perceived as more complicated. This indicates that the current traceability systems are not easy to use.

Usefulness

Usefulness or functionality represents the fit between the business needs and the functionality the solution provides (Kwon & Seo, 2013; Lee & Kozar, 2008; Seethamraju, 2015; Wu, 2011; Yang et al., 2015). Organisations are unlikely to buy software that does not provide in their needs.

The non-adoption of interoperable traceability systems at partners affects the usefulness for an actor. A traceability system does not realise its full potential when the supply chain is incomplete (Dediu, Moga, & Cristea, 2016). Some of the benefits may still be realised like good documentation in legal defence and (minor) increase in food safety incident management (Bosona & Gebresenbet, 2013). However, paper traceability can also realise these. Thus, these do not add to the usefulness.

Organisational fit

The adoption chance is influenced by how much an organisation has to change its business processes to work with the solution. Software which is in line with current business processes is more likely to be adopted than software which requires business process changes and change management (Lee & Kozar, 2008; Seethamraju, 2015; Yang et al., 2015). The organisational fit is related to the cost of a system as changes to the business process bring costs.

Business processes may require remodelling when a traceability system is adopted. Actors must to invest in software, hardware and hire personnel to implement a traceability system. Some studies indicate that companies lack skilled employees to use the traceability system (Bosona & Gebresenbet, 2013; Duan et al., 2017). This would mean that companies require to attract additional ICT staff. For those who have skilled employees: "One of the main disadvantages is that information sharing might increase the workload of staff" (Minnens et al., 2018).

Another problem is the number of available standards. The available traceability systems rely on different techniques (Paper-based, EDI, GS1 EPCIS, barcodes and XML) as well as different standards within a chosen technique: "some major buyers have their own "flavors" of EDI, forcing customisation even within a "standard."" (Dediu et al., 2016). Because each actor has different requirements for traceability systems, there is no "one size fits all" solution. For example, the use of a certain standard is favourable for one organisation as they already use it, while it may not be compatible with another organisation. Moreover, for some of the standards, it is never possible to create interoperability, as they capture other data or rely on different techniques. This happens when one company keeps track of pallets, while another keeps track of crates or individual products.

Benefits

The benefits a solution has over the current way of working. When a new solution provides minimal benefits over the old solution it is less likely to be adopted (Kwon & Seo, 2013; Wu, 2011).

A traceability system should bring benefits to an organisation. However, the return on investment (ROI) is unsure (Dediu et al., 2016). One of the reasons for the unsure ROI is the intangible benefits. A traceability system does not guarantee increased revenue. The ROI can only be estimated and can never be measured precisely because the implementation of a traceability system alters the observed environment. A traceability system saves production on contaminated goods and can save the image of an organisation or product, it results in fewer losses when a food safety incident happens. But there is no guarantee that food safety incidents occur, if they do not occur, no benefits are realised. Another cause of unsure ROI is that most of the benefits are unlikely to be realised until a critical mass has adopted a traceability system. As indicated earlier, in the usefulness factor, an automatic traceability system

realises minimal benefits when the supply chain is incomplete (Dediu et al., 2016).

Security and trust

Security and trust indicate how secure an application is and can be trusted by end-users to create value. It contains concerns about security and fear of losing control. Security and trust is a relevant factor affecting the adoption decision for cloud services (Wu, 2011; Yang et al., 2015), but not for anti-malware (Seethamraju, 2015)

Multiple studies argue that there is a lack of trust between the actors in the supply chain (Minnens et al., 2018; Storoy, Thakur, & Olsen, 2013; Bosona & Gebresenbet, 2013). However, this lack of trust originates from the sensitive nature of the data that must be shared between supply chain partners for a traceability system, not the relationships per se. Organisations may trust each other enough to work together, but not enough to share sensitive data. This is reflected by a Delphi study for adoption of traceability systems: "The majority of stakeholders consider a traceability system only promising if the data confidentiality is guaranteed by the data infrastructure" (Minnens et al., 2018).

In total there are 6 possible business risks for actors in the supply chain when sharing traceability data identified. Some of these risks can never be guaranteed to be prevented as actors can gather information through other means than a traceability system. For example, two neighbouring farmers who physically talk about the trader they both sell to.

1. *B1* and *B2* may conspire against *A*. *A* cannot sell its product for competitive prices. This Business risk was observed in America where chicken processors allegedly conspired to buy chicken below competitive prices (Wiseman, Sanderson, & Robb, 2018).

2. Similar to the previous risk, such a conspiracy can happen upstream. *UB1* and *UB2* may conspire to drive up their sales prices. In the motivating example, this would happen when farmers work together to gain more value for each cow.

3. *UB1* and *B1* may learn of the existence of each other through communication with *A*. At this moment they can estimate profit margins from *A*, and handle accordingly. Moreover, they may realise that they no longer require *A*, skipping *A* or set up a venture to exclude *A*.

4. A variation in supply or demand after a better supplier or buyer has been found through the information in a traceability system. E.g. when *A1* makes contact with new *UB*s with data extracted from *A2*. This happens when two lasagne producers compete, and one can buy tomato sauce or beef from the competitors' supplier.

5. In the case that a traceability system is linked to production data, company secrets like recipes can leak. If a competitor acquires secrets from an actor, the actor may lose the uniqueness of their product.

6. large food processors are said to produce the same food product with different labels for multiple retailers. For example, the dutch "Hema rookworst" is said to be produced at UNOX, a large producer of "rookworst" (Novum, 2007). In general, a producer which produces lower grade products for other actors next to their high-grade product may be hesitant to share their traceability data. Especially since leakage of this knowledge may result in negative branding.

Examples from the past, like the chicken example in the first risk, may make organisations hesitant to implement a traceability system. Moreover, the need-to-know culture in the industry hinders data sharing (Brewster, Seepers, & all WP Participants, 2018).

Moreover, Belgian authorities give fines when a food recall happens (Minnens et al., 2018). A traceability system would carry the information that a food recall happened. Therefore it would no longer be possible to hide recalls from authorities because the proof of the recall would be in the data. Additionally, by the nature of a traceability system, the origin of food safety incidents or adulteration is more likely to be found. Once the information is out that an incident has happened in a certain organisation, they may lose a higher market share than without the traceability system. But this argument can also be used as a benefit: the ability to blame one specific organisation prevents negative branding for the others in the sector.

Besides the business risks, there are technical risks. Although they are technical, they still have an impact on the value creation process. Technical risks for traceability systems include: vendor lock-in, unexpected scalability costs and system availability risks.

Trialability

The option for an organisation to test the product. By testing the product, organisations or end users have a chance to feel the benefits of changing to a new system (Lee & Kozar, 2008; Yang et al., 2015). This factor was not found in the literature on adoption of traceability systems.

Vendor reputation

when organisations compare software solutions, organisations look at the reputation of the service provider. The time in business of the vendor and the willingness to help in implementation and production affect the vendor reputation factor. The vendor reputation is found to be unaffected by a vendor lock-in, as the switch between different solutions would always require change management (Seethamraju, 2015).

In the agri-food supply chain, the required trusted third party to handle the traceability data reflects the vendor reputation factor (Minnens et al., 2018). A software owner with control over the data may sell traceability information for profit, which the actors would want to keep secret. Or the software owner finds trade secrets in the data.

Social factors

Social factors represent environmental pressure to adopt new solutions. These can originate from competition and trade partners (Wu, 2011; Yang et al., 2015) and internal as a status symbol (Lee & Kozar, 2008). However, some sources indicate that social pressure does not influence adoption decisions (Seethamraju, 2015).

For the agri-food supply chain, each time a food safety incident happens, social factors push to the adoption of traceability systems. This can be observed from the amount of research on traceability systems after the horsemeat scandal.

Technology

Technology can be perceived from different perspectives. On one hand interest in new technologies can increase the likelihood of adoption (Wu, 2011). On the other hand, technology can be perceived as difficult and cost time to understand. This decreases the likelihood of adoption. It should be noted that the outsourced technology risks are considered minimal by SMEs as their IT own capabilities are perceived as less than those of service providers (Seethamraju, 2015). Technology readiness can be seen as a combination of a few previous factors (benefits, ease of use, security, organisational fit and trialability) and is found to have a significant impact on SaaS readiness (Yang et al., 2015). In anti-spyware adoption, computing capacity significantly impacts adoption decisions (Lee & Kozar, 2008).

For traceability systems, technology can be one of the barriers. The technology of traceability systems is perceived as too difficult, this is related to the ease of use and lack of skilled personnel factors. Additionally, the security of sensitive data in the traceability system can become a barrier to adopt a traceability system (Minnens et al., 2018). Another technology-related problem is that traceability systems may require additional IT infrastructure not present at the supply chain actor.

Costs

Finally, in cloud service adoption, the economic factors are perceived to increase the value of cloud solutions. These factors include: applications are faster up and running, less maintenance costs and improve manageability (Kwon & Seo, 2013). On the other hand, costs do not seem to affect adoption in anti-spyware (Lee & Kozar, 2008). For SaaS adoption, the costs are seen as a benefit factor, not a separate factor (Seethamraju, 2015; Yang et al., 2015).

The costs of a traceability system are one of the most prominent barriers. One of the issues identified in multiple studies is a lack of resources available for a traceability system (Minnens et al., 2018; Storoy et al., 2013; Bosona & Gebresenbet, 2013; Dediu et al., 2016; Duan et al., 2017; Hardt, Flett, & Howell, 2017). A traceability system requires companies to invest in software, hardware and hire personnel to implement a system. Post-implementation costs may

include software licensing, hardware and employees. The lack of skilled employees means that companies would require additional ICT staff. For those who have skilled employees, it would mean that their employees are less productive because it might increase the workload of staff (Minnens et al., 2018). Additionally, actors may perceive digital systems as more expensive than records on paper. The many standards further increase the costs of traceability systems. To make two traceability systems interoperable, they require expensive custom-built interfaces.

Marketing

Although marketing does not directly influence the adoption chance, it influences social factors, ease of use and usefulness and can thereby increase adoption chances (Wu, 2011). This is an effect worth noting for solution providers. Marketing was not mentioned in the literature on perceived barriers.

3 Requirements

In design science, the problem investigation is followed by the treatment design. The first step in the treatment design is the specification of requirements. The requirements provide guidelines in the search for possible treatments and stem from the stakeholder goals (Wieringa, 2014). This section is used to answer the second research question: What requirements should a traceability system satisfy in the agri-food supply chain? The rationale is that when the requirements are satisfied, the stakeholders should not or minimally perceive the barriers to adoption. In table 1, the requirements are summarised and the concerns they address are listed.

Design science provides guidance on a meta-level, therefore the Volere requirements template was used to specify the requirements (Robertson & Robertson, 2007). Each requirement is first explained, then summarised in a table structured in line with the Volere template. The Volere requirement template specifies 13 fields for each requirement:

1. **Requirement** A unique ID for the requirement.
2. **Requirement type** The type of requirement, the available options are: functional requirement, nonfunctional requirement, project constraint, design constraint, project driver, and project issue (Robertson & Robertson, 2007).
3. **Event/use case** Events or use cases that use this requirement.
4. **Description** "A one sentence statement of the intention of the requirement" (Robertson & Robertson, 2007)
5. **Rationale** "A justification of the requirement" (Robertson & Robertson, 2007). For this, the contribution argument from design science is used. The contribution argument justifies the choice for a requirement. It is structured in the following format: "(Artifact Requirements) $\times$ (Context Assumptions) contribute to (Stakeholder Goal)" (Wieringa, 2014).
6. **Originator** The stakeholder that raised this requirement.
7. **Fit criterion** A method to test if the requirement is satisfied.
8. **Customer satisfaction** Indicates the happiness of stakeholders once this requirement is satisfied. The score ranges from one to five, one indicates uninterested, five extremely pleased. As argumentation Kano's model has been used (Spool, 2019). Kano's model defines three categories of satisfaction generators: the excitement generators, the performance payoff, and basic expectations. Excitement generators are features in a product that are new to the customer. Because they are new and exciting, they create satisfaction for the customer. For example, free WiFi in a hotel when it was new. However, as time progresses, and more organisations provide the same excitement generator, it becomes a basic expectation.

Basic expectations do not generate satisfaction, they generate dissatisfaction when absent. In the WiFi example, consumers expect free WiFi in a hotel. It does not bring consumers joy when it is present. However, it brings dissatisfaction when there is no WiFi. The performance payoff is a linear satisfaction factor. The more pieces of it present, the more satisfaction it generates. An example of this is the number of functions a text editor has, more functions mean more excitement for the consumer.

9. **Customer dissatisfaction** Indicates the unhappiness of stakeholders if this requirement is not satisfied. Customer dissatisfaction is scored on a one to five scale, where one indicates hardly matters and five extremely displeased. As argumentation, Kano's model was used (Spool, 2019).

10. **Priority** "A rating of the customer value"(Robertson & Robertson, 2007). To quantify the priority, the MoSCoW model has been used (O'Connor, 2016).

11. **Conflicts** If someone implements this requirement, it is not possible to implement the following requirements.

12. **Supporting materials** A pointer to other documents that explain and illustrate the requirement. This was not used in this document, it suites architectures of multiple files and folders better.

13. **History** "Creation, changes" (Robertson & Robertson, 2007). Was not used in the specification of the requirements.

3.1 Traceability

In the introduction, one of the identified root causes was a lack of traceability. To solve the lack of traceability, multiple traceability systems have been proposed. The traceability requirement represents the demands to solve the lack of traceability. Besides the creation of traceability, a traceability system may solve other root causes as well. For example, a traceability system could influence the long and complicated supply chains, because it grants vision on the supply chains.

3.1.1 Traceability - data

By the European law (Regulation (EC) No 178/2002), food processors are required to keep: information of any person from whom they have been supplied or have supplied with food and are capable of making this information available to authorities. This applies to all stages of production, processing and distribution. To comply with the law, each supply chain actor should keep records of:

- Organisations to whom they have sold their product. There is no restriction on the size of a consignment. Records may be kept for individual products or a truckload.

Requirement	**Related concerns**
Traceability - data	Data are required for traceability and are required by law to be available (Regulation (EC)No 178/2002).
Traceability - linked	When traceability records are separated, it costs a lot of effort to match the right records.
Traceability - query	Each food safety incident has different clues to start the search for the origin of the contamination and many different questions could be asked to the traceability system.
Traceability - backup	Some actors disappear to make traceability of their counterfeit products impossible(Flari et al., 2014).
Access control	Traceability systems handle sensitive data(Minnens et al., 2018). A need-to-know culture when data are shared (Solanki & Brewster, 2014). Is the data safe? Part of the traceability data must be accessible to achieve traceability.
Value proposition - compelling feature	Current traceability systems have subjective gains. Current traceability systems only realise value when the entire supply chain adopts a traceability system
Value proposition - easy to use	Supply chain actors lack skilled employees to use traceability systems.
Value proposition - data reuse	Worries about additional workload when a traceability system is adopted(Minnens et al., 2018). The system should replace or reuse current systems, otherwise, it is an additional burden (Minnens et al., 2018). Traceability systems require a lot of investment.
Trusted software supplier - supplier reputation	Supply chain actors require a trusted third party to handle the traceability data (Minnens et al., 2018).
Trusted software supplier - reliable system	The system should work properly. The system should not create a vendor lock-in.
Trusted software supplier - governance	Supply chain actors require a neutral governance body they can trust.

Table 1: Requirement - concern relationships. This table lists which concerns are addressed by a requirement.

- Organisations from whom they bought their products.
- A record of all the ingredients in a sold product.

This obligated record-keeping enables traceability. Additionally, this should realise the mentioned benefits of traceability. Actors use the captured traceability data to find the origin of a food safety incident. Once the origin is found, actors and food safety authorities use the traceability data to find where other contaminated goods went to, to prevent human ingestion, save the image of organisations, and prevent waste production.

This is a functional requirement because it is the goal of a traceability system. The rationale for this requirement is: If the system can capture traceability data, and assuming that all supply chain actors cooperate, then the system contributes to solving part of the root causes of food safety incidents. This requirement originates from the regulations laid down by food safety authorities, the demand for safe food of consumers, and agri-food supply chain actors who want to realise the benefits of traceability (section 1.2). This requirement is satisfied when actors can find the origins and destinations of their products. The customer satisfaction is graded on a 2. Because officially all organisations are supposed to have traceability in place. It is a basic expectation that will not increase customer satisfaction. Because traceability is a basic expectation, absence of this feature will displease organisations. Customer dissatisfaction is therefore graded with a 5. Given the importance of this requirement, the priority is a must. This requirement does not conflict with other requirements.

3.1.2 Traceability - linked

Fast traceability requires more than capturing data. Currently, all supply chain actors should capture traceability data because it is required by law. But as shown in the introduction, this is not fast enough. At this moment, the Regulation only obliges actors the capability to identify any person(UB) from who A, has bought a food or feed product and vice versa (Regulation (EC) No 178/2002). However, this does not mean that the exact product can be named. For example, when a cattle farmer sells beef to the lasagne producer. The lasagne producer keeps a record that he bought beef from the farmer. The farmer keeps a record of sold beef. This will work for one product or batch, but when the amount of consignments increases, it will be hard to find the matching products, especially when different record types are kept (trolleys v.s. crates). When actors share an identifier of the products, they can safe time to find the matching product. A possible implementation of this requirement could look like: actors share the barcode of a product. For example, if a lasagne producer keeps a record of the barcodes on their ready-made lasagne and the supermarket keeps the barcodes as well, the supermarket can indicate exactly which products are affected. This limits the number of records to search through at the lasagne producer. When this is done digitally, it would enable the possibility to query through the supply chain. This would decrease the time used to find actors in an affected supply chain.

Table 2: Traceability - data

Requirement	Traceability - data
Requirement type	Functional
Events/use cases	Discovery of contamination origins, the current location of contaminated goods.
Description	The system should be can capture traceability data.
Rationale	If the system is can capture traceability data, and assuming that all supply chain actors cooperate, then the system contributes to solve the lack of traceability.
Originator	food safety authorities, consumers, food supply chain actors
Fit criterion	An actor should be capable to find the origin and destination of products.
Customer satisfaction	3.
Customer dissatisfaction	5.
Priority	Must
Conflicts	-

This is a functional requirement because it creates a solution for the long time to find the root cause of food safety incidents. Actors and food safety authorities use the links between product records to find the origins and destination of products. Because it enables querying through the different records of the data, regardless of the location. The following contribution argument can be made: If the system has links between products spanning company borders and assuming that actors need rapid access to traceability data of partners, then the product links contribute to the goal of faster contamination location discovery. This requirement originates from consumers and food safety authorities, in their request for safe food. The supply chain actors are also originators because of the benefits faster traceability brings. This requirement can be tested by looking if an actor is capable of pointing to an exact product record at a partner organisation. Because this is most likely new to supply chain actors, this is an excitement generator. As this requirement is an excitement generator the customer satisfaction is graded with a 4. Because it is an excitement generator, the absence is not likely to cause dissatisfaction and therefore graded with a 2. This requirement is a must because increasing speed in traceability is important. This requirement does not conflict with other requirements.

3.1.3 Traceability - query

A quick response to food safety incidents requires the power of querying. This is because a food incident usually has vague clues to what a possible origin was. Previous research has demonstrated this with six example knowledge questions that could be asked to a traceability system to help solve a food safety incident (Solanki & Brewster, 2014). These questions have a variety of entry points(E.g. data at a retailer, data of consumer organisations, and food producers) or clues to a location to start the search for a food safety incident. The ability to query the data creates freedom to search and order the data in a structure that contributes to finding clues and root causes of a food safety incident, rather than an endless search for a needle in a haystack of data.

This requirement is a functional requirement because it creates a function for the system. The ability to query is used to search for clues of the root cause for food safety incidents. The requirement can be described as: The system must support queries on the traceability data. The rationale behind this requirement is that when a system can query trough the traceability data, then the ability to query contributes to the goal of faster contamination location discovery. This requirement originates from the food safety authorities, consumers, and supply chain actors, who are concerned with food safety. This requirement is satisfied when an actor is can query trough the traceability data. This requirement is a basic expectation because almost every system can query. Therefore, the customer satisfaction is graded on a 2 and the customer dissatisfaction on a 5. Because it is a basic expectation, this requirement is a must. Unfortunately, this requirement could conflict with the concerns of other requirements: access control and value proposition - easy to use. To query data of multiple other actors, an actor requires access to data of others. Which conflicts with the concern

Table 3: Traceability - linked

Requirement	Traceability - linked
Requirement type	Functional Requirement
Events/use cases	To find the originator (producer) of the contaminated product.
Description	The system should have links between products in different organisations at the level of individual batches or consignments.
Rationale	If the system has links between product spanning company borders and the system can be queried and assuming that actors need rapid access to traceability data of partners, then the product links contribute to the goal of faster contamination location discovery.
Originator	Food safety authorities, consumers, food supply chain actors
Fit criterion	An actor can of point to an exact product record at a partner organisation.
Customer satisfaction	4.
Customer dissatisfaction	2.
Priority	Must
Conflicts	-

Table 4: Traceability - Query

Requirement	Traceability - query
Requirement type	Functional Requirement
Events/use cases	In the search for clues to a root cause of food safety incidents.
Description	The system must support queries on the traceability data.
Rationale	If the system can query the traceability data, then the ability to query contributes to the goal of faster contamination location discovery.
Originator	food safety authorities, consumers, food supply chain actors
Fit criterion	An actor is can query trough the traceability data.
Customer satisfaction	2.
Customer dissatisfaction	5.
Priority	Must
Conflicts	-

of actors regarding the sensitivity of the data. A solution that shares specific parts of the data rather than an entire dataset, can address the concern for data safety and the demand for queries. For example, a tomato sauce producer has no reason to hide that he uses tomatoes, as this is clear from the actors' role. End users could perceive the creation of queries as difficult. However, software developers usually automate queries in the applications for end users. Therefore, the ability to query should not conflict with the easy to use requirement, as end-users do not see queries. When a new question arises, a software developer can add this as a new functionality.

3.1.4 Traceability - backup

The fourth requirement is a backup mechanism. The previous requirements have the flaw that once an actor disappears, removes his data, or is unavailable, traceability is impossible. This became apparent in the 2011 organic food crisis (Flari et al., 2014). In this crisis, fraudsters opened and closed short term organisations to hamper traceability. Therefore a mechanism should be present to recreate traceability without the cooperation of an actor. The backup can create a trust-related issue because organisations may not trust the party responsible for the backup. Because the backup is out of control of the actors, actors no longer possess the opportunity to choose who can access their data.

This requirement is a functional requirement because it creates a new function for the system. The backup mechanism reconstructs the traceability when

Table 5: Traceability - backup

Requirement	Traceability - backup.
Requirement type	Functional requirement
Events/use cases	When an actor disappears or does not share data when obliged.
Description	A backup mechanism should be in place to recreate the traceability when actors disappear.
Rationale	If the system has backup traceability, assuming that some actors try to hide their criminal activities, then the backup contributes to the goal of preventing data loss.
Originator	Food safety authorities.
Fit criterion	An actor should be capable to find the origin and destination of a product when some of the actors are missing.
Customer satisfaction	2 & 5.
Customer dissatisfaction	2.
Priority	Should
Conflicts	*Access control*,

an actor does not cooperate. The contribution argument for this requirement is: if the system has backup traceability, assuming that some actors try to hide their criminal activities, then the backup contributes to the goal of preventing data loss. This requirement originates from food safety authorities because they can not prosecute the involved actors and can not ensure food safety. The satisfaction of this requirement can be tested by an actor who can find the origin and destination of a product when some of the actors are missing. For food safety authorities this generates a lot of satisfaction because it helps them in their job. Therefore the customer satisfaction is graded a 5. The customer dissatisfaction is graded a 2, Because this it is new for food safety authorities to have a second location where traceability data can be retrieved, the absence of it is not expected to generate high dissatisfaction. This requirement is given a should because it creates benefits for the food safety authorities and possibly contributes to food safety, but does not directly reduce the barriers to adoption for the supply chain actors. This requirement could conflict with the access control requirement.

3.2 Access control

As indicated in the previous section, traceability systems fail due do lack of trust. "A system is only considered promising if data confidentiality is ensured on a technical level" (Minnens et al., 2018). This requirement represents concerns of stakeholders and the security and trust factor(section 2.3). Therefore, an access control framework should be in place. An access control framework can help in the prevention of a large part of the risks as well as address the stakeholder concerns about the safety of the system. Specific access control could mitigate the 6 risks of the security and trust factor. Specific access control achieves this by sharing data required for traceability (sourcing of ingredients) but keeping transaction records private. Moreover, the culture in the agri-food supply chain prevents data sharing between actors more than one up or downstream (Brewster et al., 2018). However, actors require lower trust in each other when they can share specific parts of their data. E.g. when two parties trade a physical good, they can safely share the bar-code (or another identifier) of that product because both locations know this information. This reduces the trust required between parties to share part of their data. The use of an access control framework is expected to lower the risks posed by the sensitive nature of the data and thereby the chances of adoption are increased.

This requirement represents a functional requirement because it is essential for the system to have, to be considered promising by actors. The access control requirement is used when an actor tries to gather traceability data from other actors in the supply chain. For this to happen in a manner that aligns with the stakeholder concerns, the system should allow actors to define who has access to parts of their data. Because actors do not join systems where their data safety is not guaranteed, the possibility for actors to specify who has access to their traceability data contributes to the adoption of a traceability system. This requirement originates from the supply chain actors who worry about the sensitive nature of their data. This requirement is satisfied when data cannot be accessed by other actors who do not have access. Customer satisfaction is scored on a 2. This requirement will not contribute to the satisfaction because it is a basic expectation. Because it is a basic expectation, the dissatisfaction will be high and therefore scored a 5. The concerns of the stakeholders indicate this requirement is a must as they do not adopt a system without access control.

3.3 Value proposition

Because the costs, organisational fit and unsure benefits are prominent barriers to adoption the value proposition attempts to lower these barriers. In contrast to the traceability requirements, the value proposition requirements are present to lower the barriers to adoption. The value proposition requirements relate to the factors ease of use, usefulness, organisational fit, benefits, and costs.

Table 6: Access control

Requirement	Access control
Requirement type	Functional requirement
Events/use cases	An actor wants to see data of a partner.
Description	Each actor can define who has access to which part of their data.
Rationale	If the system has the possibility for actors to specify who has access to their traceability data, assuming that actors do not join systems where their data safety is not guaranteed, then the access control contributes to the adoption of traceability systems.
Originator	Supply chain Actors.
Fit criterion	Data cannot be accessed by unauthorised actors.
Customer satisfaction	2.
Customer dissatisfaction	5
Priority	Must
Conflicts	*Traceability - backup*

3.3.1 Value proposition - data reuse

The design decisions in a traceability system influence the costs to implement a traceability system. Because resources are tight for a large proportion of the supply chain, large investments would mean hurdles for a large proportion of the supply chain. One of the possible ways to lower the costs of traceability systems is to increase the organisational fit factor. minimising the amount of business process remodelling lowers the costs of adoption. This can also be seen in the demand for systems to reuse or replace current applications instead of being an additional one (Minnens et al., 2018). The possibility for traceability systems to reuse the data from existing systems is expected to decrease the adoption costs as the reuse of data from current applications limits the amount of business remodelling. Additionally, it helps in the ease of use.

This requirement is a design constraint because it imposes restrictions on how the system should be designed. This requirement is used when the system loads data from other existing systems to create traceability. This requirement can be described as: the system should reuse existing data where possible. The rationale behind this requirement is that if the system reuses existing data where possible and the reuse of data lowers the implementation costs, then the reuse of data lowers a barrier to adoption. This requirement originates from the supply chain actors, that lack the resource required to implement a traceability system. Satisfaction of this requirement can be tested by the ability to import data from existing systems. This requirement realises a payoff satisfaction because the more data it can import, the more satisfaction is generated. Therefore the customer satisfaction is scored on a 3. The dissatisfaction has been scored on a 3 as well. This requirement has a priority of should because it contributes to the adoption of the system. However, it can function without it and can be added later. This requirement may conflict with value proposition - easy to use, because integration with other systems may create a difficult system for end-users. Moreover, a system that relies on legacy systems could worsen future adjustments to the system or application landscape of actors. Reusing data can save manual work and contribute to the easy to use requirement.

3.3.2 Value proposition - compelling feature

One of the encountered problems in the barriers of adoption is that the entire chain must adopt the traceability system to realise value. A single supply chain actor, that does not adopting the system, breaks the business case for the entire supply chain. To prevent this, the system should have a positive business case independent of the number of users. A new business process which generates value and creates traceability should create a positive business case. This requirement does not propose a specific compelling feature. At the time of requirement specification, the author did not know how to satisfy this requirement yet. Changing it to a specific feature would conflict with the search for possible solutions as well as limit the creativity for possible solutions.

In a Delphi study, 80% (30 participants) indicated to adopt a traceability

Table 7: Value proposition - data reuse

Requirement	Value proposition - data reuse
Requirement type	Project constraints
Events/use cases	Adding data from other systems.
Description	The system should reuse existing data where possible.
Rationale	If the system reuses existing data where possible, assuming that the reuse of data lowers the implementation costs, then the reuse of existing data increases the chances of adoption.
Originator	Supply chain actors
Fit criterion	Most of the data are imported.
Customer satisfaction	3.
Customer dissatisfaction	3.
Priority	Should.
Conflicts	Easy to use

system direct or as early adopters (Minnens et al., 2018), this number is expected to be lower in reality for two reasons: the first reasons is due to their self-selected sample. By the self-selection, companies interested in traceability systems are more likely to participate in the study than companies who are not interested. Therefore a warning is made for generalisation outside the study sample. Second, it is easy to say that you will adopt a system, but doing it in practice takes more effort. Therefore, 80% is expected to be an overestimation of reality. Moreover, even a percentage of 80% would imply that in a supply chain of five actors, one actor would not adopt a traceability system, damaging the business case.

The requirement type is a project driver because it is a business-related force that drives the adoption of a traceability system. The requirement can be described as: the traceability system should have a feature that brings value for adaptors in itself when no other actors are adopting. The rationale for this requirement is that current traceability systems do not realise value when other actors do not adopt the traceability system, a compelling feature should contribute to the value creation of the traceability system. This requirement originates from the supply chain actors because they perceive the lack of others joining the system as a barrier. A compelling feature is expected to be an excitement generator because it is most likely new for the sector. Because it is an excitement generator, the customer satisfaction is graded on a 5, and the customer dissatisfaction is graded on a 2. The priority of this requirement is graded on a should, because it is not required for a traceability system to function and can be added later. But it helps in lowering the barriers to adoption. This requirement does not conflict with other requirements.

Table 8: Value proposition - compelling feature

Requirement	Value proposition - compelling feature
Requirement type	Project driver
Events/use cases	
Description	The system should have a feature that realises value even when a single actor adopts a traceability system.
Rationale	If a traceability system has a compelling feature and assuming that current systems do not realise value when others do not adopt the traceability system, then the compelling feature contributes to the perceived value.
Originator	Supply chain actors.
Fit criterion	A new feature apart from traceability that brings value.
Customer satisfaction	5.
Customer dissatisfaction	2.
Priority	Must/should.
Conflicts	-

3.3.3 Value proposition - easy to use

This requirement represents the concerns of end users, supply chain actors and the ease of use factor. To counter the lack of skilled employees, concern for lower efficiency and possible easier paper traceability, traceability systems should be easy to use. The system should prevent the need for (new) skilled employees because additional employees lower operational efficiency and increase the perceived bureaucratic burden. Moreover, an easy to use system should reduce the concerns of end users about their capability to use the systems. Traceability systems which are easy to use should not require additional personnel and may even be easier to use than a paper-based version. Moreover, based on the ease of use adoption factor, an easy to use system receives larger uptake. Unfortunately, the perception of individuals determines satisfaction of this requirement. E.g. LaTeX and Microsoft Word are both text editors, but the former can be perceived more as a burden to edit text than the latter. Marketing and trialability factors may help in this regard to demonstrate potential value and ease of use. This requirement gains importance towards the end of the implementation when the user interface is created. This requirement should still be considered in the design of a solution to prevent a lock-in to complicated solutions.

This is a non-functional requirement because it poses restrictions on the quality of the traceability system. This requirement is seen in the daily use of traceability systems and in the marketing of a system. The requirement can be described as: the system should be easy to use for end users. This rationale behind this requirement is that if the system is easy to use and assuming that the end users understand how the traceability system works, then the easy to use requirement contributes to the perceived value of a traceability system. By increasing the perceived value of a traceability system, it lowers the barrier where traceability systems are perceived as difficult to use. Additionally, an easy to use system could prevent the need for additional employees to use the system for supply chain actors. This requirement originates from the agri-food supply chain actors and the end users. Because the end users had concerns if they were skilled enough to use the system, might lose their job because of it, or would have a higher workload. The supply chain actors are originators because they are concerned about the productivity of their employees. The customer satisfaction is graded on a 3 because this requirement is believed to be a performance payoff. Because with more investment in features, the system becomes easier and easier to use. The customer dissatisfaction is graded on a 5 because a complicated system is unlikely to be used by actors because they do not understand the value it brings. The priority is graded on a could. Although its importance, this requirement could be satisfied in later stages when a user interface is updated to better fit this requirement. Easy to use is not affected by other requirements.

Table 9: Value proposition - easy to use.

Requirement	Value proposition - easy to use.
Requirement type	Nonfunctional requirement.
Events/use cases	daily use of traceability systems
Description	The system should be easy to use for end users.
Rationale	If the system is easy to use and assuming that the end users understand how the system works, then the easy to use requirement contributes to the perceived value of a traceability system.
Originator	Agri-food supply chain actors, end users.
Fit criterion	The perceived ease of use by end users.
Customer satisfaction	3.
Customer dissatisfaction	5.
Priority	Could.
Conflicts	-

3.4 Trusted software supplier

The trusted software supplier is a requirement category of three requirements for the creator of traceability systems. It is related to the security and trust factor, the vendor reputation factor and the concerns regarding the sensitive data of supply chain actors.

3.4.1 Trusted software supplier - supplier reputation

The industry requires a trusted and neutral third party to handle the data because of the sensitive nature (Minnens et al., 2018). The supply chain actors should trust the company that builds traceability systems. The software supplier should preferably not have large ties with large food corporations. Moreover, the trusted software supplier may have access to the data, or own the data depending on the implementation. But the software owner can also outsource the storage of the data. In any case, the supply chain actors should trust the trusted software supplier. Especially the software supplier should be trusted not to sell the data or inferences without the consent of the actors. Trust in the software supplier should increase when they are neutral. Additionally, government agencies are not an option because they are not perceived as neutral by the industry (Minnens et al., 2018). For later stages, a vendor who helps actors during implementation of traceability systems is expected to contribute to the uptake.

This requirement is a project issue requirement because it defines a condition for traceability. Some events of this requirement are: handling of sensitive data, during the adoption of a traceability system, and when actors experience problems with a traceability system. This requirement can be described as: the supplier should have a good reputation and can be trusted with the sensitive data of organisations. The reason for this requirement is that actors do not buy systems where their data are not save. Moreover, as actors have experienced difficulties in the implementation of traceability systems, a vendor that helps them in their implementation. Therefore, a good supplier reputation is expected to lower a barrier to adoption. This requirement originates from the agri-food supply chain actors that are concerned for their sensitive data and for possible difficulties during implementation. A fit criterion for this requirement can be a survey under adopters for their opinion on the software supplier. A vendor with a good reputation is expected to be a basic expectation. Therefore it is expected to generate minimal excitement and scored on a 2. The customer dissatisfaction has been graded on a 5, because of the sensitive nature of the data handled. Organisations are not expected to join a system which is owned by an organisation with a negative reputation on data safety. Moreover, the dissatisfaction is expected to be high when a basic expectation is not present. This requirement **should** be satisfied because it creates a significant barrier for actors before adoption but does not have a direct effect on a traceability system itself. Additionally, the reputation of a software supplier is fluid and can be affected after the creation of a system. But, a reputation is easier to diminish than to build. This requirement does not conflict with other requirements.

3.4.2 Trusted software supplier - reliable system

Additionally, the actors require a software supplier that builds a reliable system that functions properly. Moreover, the system should not generate a vendor lock-in.

This requirement is a nonfunctional requirement because it creates a quality feature for the system. Ideally, actors should never encounter this requirement, because that would mean that it always works. And if it does not work properly, actors can switch to another traceability system with minimal effort. This requirement can be described as: the system should work appropriately and not have a lock-in. The rationale behind this requirement is that actors are unlikely to adopt systems that do not function properly or have a lock-in. When a system functions properly and does not have a lock-in, actors perceive a lower barrier to adopt such a system. This requirement originates from the agri-food supply chain actors, who have limited available resources. When a system does not function properly, it hurts their value creation and the ROI of the system. Moreover, a lock-in raises prices for actors to switch to another system, even when they would want to switch. The up-time of the system can be used to test satisfaction of this requirement. This requirement is judged as a basic expectation because actors expect a system to function when they buy into it. Therefore the satisfaction is graded on a 1 because it does not bring joy

Table 10: Trusted software supplier - supplier reputation.

Requirement	Trusted software supplier - supplier reputation
Requirement type	Project issue.
Events/use cases	Handling sensitive data, Adoption of software, Help during events.
Description	The supplier should have a good reputation and can be trusted with the sensitive data of organisations.
Rationale	If the supplier has a good reputation, and assuming that organisations are unlikely to adopt software from organisations who do not handle sensitive data appropriately and software is more likely to be bought from software vendors known to help their customers in implementation and problem solving, then the supplier reputation helps in the adoption chance of traceability systems.
Originator	Agri-food supply chain actors.
Fit criterion	A survey at the adopters of a traceability system.
Customer satisfaction	2.
Customer dissatisfaction	5.
Priority	Should
Conflicts	-

Table 11: Trusted software supplier - reliable system.

Requirement	Trusted software supplier - reliable system.
Requirement type	Nonfunctional requirement
Events/use cases	When the traceability system is not available.
Description	The system should work appropriately and not have a lock-in
Rationale	If the system is reliable, and assuming that actors are unlikely to adopt a system that is known to be unavailable a lot of the time, then the reliable system helps in the adoption chance of a traceability system.
Originator	Food supply chain actors.
Fit criterion	The amount of time the system is available.
Customer satisfaction	1.
Customer dissatisfaction	5.
Priority	Must.
Conflicts	-

when it functions as it should. The dissatisfaction is graded on a 5 because consumers are expected to be extremely unhappy when the system does not function properly. Implementation of this requirement is a must because an unreliable system will cause the system to be a failure from the start. This requirement does not conflict with other requirements.

3.4.3 Trusted software supplier - governance

Every system will require some sort of governance. As time passes environments change, and so will traceability systems. As time progresses new laws may require the capture of new data or changes to a data model. Because of these inevitable future changes, a governance model or governance organisation is required. The governance body should be neutral and can be expected to be in existence for the foreseeable future "particularly given the short life span of many software companies" (Seethamraju, 2015).

This requirement is a project issue because it defines a condition for the traceability system to function within. This requirement is used in future adaptions or updates of the traceability system. This requirement can be described as: the traceability system should have a neutral governance structure. The rationale for this requirement is that changes to a system are inevitable. To cope with these changes, a governance structure is required. Similar to the supplier

Table 12: Trusted software supplier - governance.

Requirement	Trusted software supplier - governance
Requirement type	Project issue.
Events/use cases	Future developments.
Description	The created system should have a neutral governance structure
Rationale	If the traceability system has a neutral governance structure, and assuming that the system will require changes in the future, then actors are more likely to adopt the system.
Originator	Food supply chain actors
Fit criterion	A neutral governance body or organisation is present.
Customer satisfaction	2.
Customer dissatisfaction	5.
Priority	Would.
Conflicts	-

reputation, the governance body should be trusted by the supply chain actors. food safety authorities are said to play an important role but are also not trusted by the industry (Minnens et al., 2018). Therefore food safety authorities could be part of a governance body, but not be the only governance actor. The availability of a neutral governance body is expected to lower barriers to adopt a traceability system. This requirement originates from the supply chain actors. The satisfaction of this requirement can be tested to look if a neutral governance body is present. Governance of a system is a basic expectation. Therefore the satisfaction is graded on a 2, and the dissatisfaction on a 4. The priority of this requirement is a would, because it does not affect the development of a system and can be satisfied in later stages of development. This requirement does not conflict with other requirements.

3.5 Out of scope requirements

One of the expressed concerns by some of the supply chain stakeholder is that data must be correct. Although some solutions may exist which can predict faulty data, this is not the goal of this research and is therefore excluded.

Another issue is the cost structure. Because only thirty per cent of the industry stakeholders are prepared to pay for a traceability system (Minnens et al., 2018), the software developer should finance the system in another way than through direct payment.

4 State of the art systems

After specifying the requirements the next step in treatment design is the search for available treatments. In the search for possible treatments two categories have been made: available traceability systems and approaches where traceability systems could be built upon. After an introduction of the traceability system, they are compared to the requirements defined in the previous section. After all traceability systems have been analysed, a gap analysis is done. Similar to the first part, the approaches are introduced, checked if they fit the requirements, and a gap analysis is done. This section ends with an analysis of ontology based access control as an access control framework.

4.1 traceability systems

As previously noted, multiple initiatives try to enable traceability. Below is a small list of traceability initiatives that were suggested, found on google and google scholar or encountered during this thesis. This list is likely to be incomplete and traceability systems are missing. The satisfaction of the requirements is summarised in table 13. The traceability systems are scored on the satisfaction of the requirements. They are scored on a three-point scale: not satisfied or not possible (-), neutral (+/-) and satisfied (+).

4.1.1 Linked pedigrees

Linked pedigrees(LP) contributes to the integrity of the agri-food supply chain in two ways. The first contribution is in the formalisation of EPCIS and CBV (explained later) into formal ontologies. Next to that, LP provides a setup for a decentralised architecture for traceability in the agri-food supply chain which can be seen in figure 2 (Solanki & Brewster, 2014). LP is elaborated in three components: the data model, the application architecture and the communication architecture.

The data model has been build on the idea of a classical pedigree which is "a record that traces the ownership and transactions of a product as it moves among various trading partners" (Solanki & Brewster, 2014), the classic pedigrees can both be paper-based and electronic. In LP, a pedigree is made for every product or batch of products. Each pedigree represents the actions an organisation has done to create their final product from ingredients. Every actor has control over their pedigrees and a privately owned storage for their pedigrees. In figure 2 this is represented by the separated actors and their individual data stores. The pedigrees are saved in linked data(RDF) format. A pedigree is identified with a URI. The pedigree is used by actors to describe the product, transaction, and consignment information. Moreover, each pedigree contains URIs to the pedigree datasets of upstream and downstream actors. For example, consider a simplified supply chain for the production of ready-made lasagne consisting of a beef trader, a sauce producer and a lasagne producer. The beef trader has a pedigree that contains the information on the sourcing of the beef,

storage conditions, crates in the consignment, and the pedigree URI of the sauce producer. The sauce producer has a pedigree that contains information on the sourcing (among other things the pedigree URI of the beef trader), production processes, and the pedigree URI of the lasagne producer. The lasagne producer has a pedigree that contains information on the sourcing (among other things the pedigree URI of the sauce producer), production processes, and information on the supermarkets. The information of these actors combined creates the traceability required. In figure 2, the purple arrows represent the flow of the food consignments (beef, sauce and lasagne in the example), while the arrows between the actors represent the exchange of pedigree URIs. To fill the pedigree with the necessary data for traceability, the OntoPedigree ontology design pattern has been made. OntoPedigree can be used to design domain-specific ontologies. Two of the domain-specific models that are incorporated in the OntoPedigree and explained below are the Electronic Product Code Information Services(EPCIS) and core business vocabulary of GS1. Initially, GS1, an organisation responsible for the barcode standards, had created barcodes and RFID tags to create traceability. However, as time progressed they kept developing standards to create traceability and information sharing. Combined, EPCIS and CBV create specifications to create traceability.

- **EPCIS** EPCIS is used to share information on product movement and status between actors. It focuses on the capture of what, where, when and why of the product to create correct and specific product information (GS1, n.d.). This data are automatically generated by the scanning of barcodes and RFID tags during production. Each production step or product movement is abstracted as an event.

- **CBV** CBV is meant to be used in combination with EPCIS. CBV provides a standard to fill the data fields defined in the EPCIS standard. The goal of CBV is to grant a common way for organisations to interpret data, which may be interpreted differently without the standard. (GS1, n.d.)

Each LP application has three main components. The Linked Data Repositories(LDR) include the data of supply chain partners and supply chain operations. Additionally, it may contain data on the products to be shipped and locations. Second, the Event Extraction Engine is responsible for extracting data from the LDR using the LinkedEPCIS library. The Linked pedigrees Generator component is responsible for generating pedigrees conforming to the OntoPedigree vocabulary and are stored in the Linked pedigrees repository. The Linked Pedigree Generator is part of the Linked Pedigree Manager agent, which also has the role to create a connection with the upstream and downstream actors, confirming the sending and receiving of goods at an actor. The LDR and Linked pedigrees repository are separate entities.

The communication architecture has two different actors, the individual supply chain actors and the Integrated Linked pedigrees Store (ILPS). Pedigrees URIs are sent in a pull model. The chain begins with an actor sending a physical consignment to a downstream actor. Upon the arrival of the consignment at

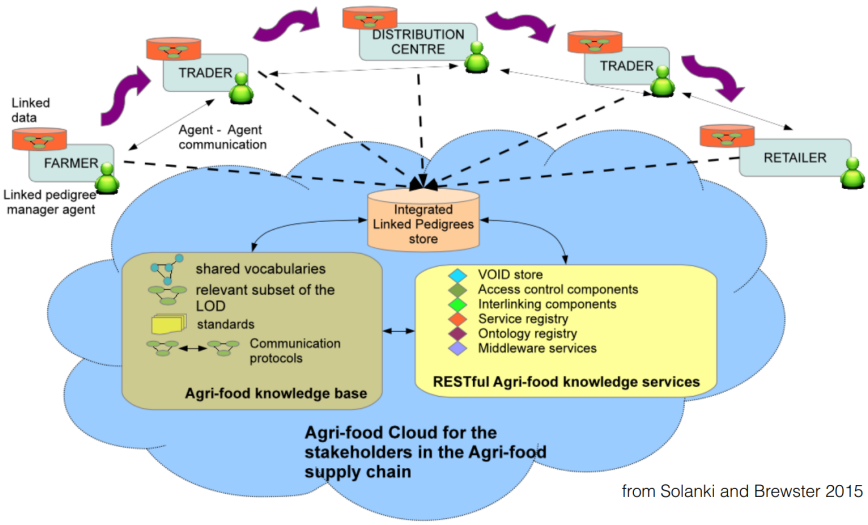


Figure 2: High-level architecture of Linked pedigrees.

the downstream actor, the downstream actor sends a request for the pedigree URI. The downstream actor responds by sending the URI to their corresponding Pedigree. At the same time, the URI is sent to the ILPS. Each actor incorporates the URIs of previous actors in its pedigree. This process is repeated until the end of the supply chain is reached. At this point, the final information and the end-of-supply-chain message is sent to the ILPS. By sending the URIs to the ILPS, integrity of the traceability chain is ensured when actors are unavailable for any reason. The ILPS contains a knowledge base with standards, communication protocols, vocabularies and a set of relevant open data. Moreover, it contains knowledge services for applications for individual parties to connect to.

Requirements checklist

Traceability - data: Satisfied, LP captures the required data for traceability.

Traceability - linked: Satisfied, during the pedigree exchange protocol, the links between the pedigree datasets of upstream and downstream actors is made.

Traceability - query: Satisfied, the use of semantic web standards creates the opportunity to create queries that span multiple triple stores.

Traceability - backup: Satisfied, the ILPS provides the ability to indicate which actor is not sharing the required information. Furthermore, it enables the reconstruction of the supply chain when an actor disappears even though it does not have actual data inside a pedigree.

Access control: Not satisfied, LP does not provide access control to data. A note is made on the abstraction of it. It should be noted that LP is designed in a manner that allows data owners to be in control of their data. The communication protocol only requires actors to send URIs of the pedigree, no supplementary data. Actors can decide who has access through the pedigree URI.

Value proposition - compelling feature: Neutral, LP does not provide a compelling feature. When only a single actor adopts LP he is not likely to realise value. The ILPS provides an access point for other application of actors. These applications may realise the value generated by LP when no other actors in the supply chain are present.

Value proposition - easy to use: Satisfied, although it is hard to judge if a system is easy to use from the architecture, the vision seems easy in use for end users. A pedigree is automatically generated each time a product arrives and linked to the sender. This creates minimal manual effort in generating traceability data when consignments arrive. One scan and the product is confirmed arrived, registered and the pedigree URI is requested from the sender. The organisation only has to incorporate a link to the received pedigree in its product pedigree when selling a product.

Value proposition - data reuse Satisfied, LP creates a loose application, that reuses data from other datastores already present at the actors to fill the pedigrees. In that way, users extract data and put it in a pedigree, rather than re-entering the data. Moreover, the pedigree URIs of up and downstream actors provide access points for potential supplementary data, assuming they are granted access.

Trusted software supplier - supplier reputation: Neutral, LP does not have a software supplier building the system. Therefore no statements can be made on the supplier regarding their behaviour with sensitive data and help in implementations. However, the architecture prevents the need to trust the software supplier with data as they are kept local. The ILPS presents a risk factor because they possess all the connected URIs of the actors at a single location. This is because the pedigree URIs point to access points of the data. These access points could be local computers of the supply chain actors. Another possibility is that the URIs contain company names. Therefore, it can be devised who is trading with who, by looking at the URIs at the ILPS. When the software supplier acts as the ILPS, the supplier must be trusted with the data again.

Trusted software supplier - reliable system: Satisfied, it cannot yet be judged if the system will always work, this will be seen when the system is in use. However, the ILPS provides a second route when B is not available or has problems with LP. Then A can use the ILPS to receive information from C. LP does not generate a vendor lock-in by itself. However, the

users of LP are locked-in to the environment, as the current architecture does not have interoperability built-in with other traceability systems.

Trusted software supplier - governance: Not satisfied, since there is no software supplier at this moment no statements can be made on the application part regarding the governance. However, the used standards in OntoPedigree, are under the governance of GS1. Moreover, the linked data structure is governed by the World Wide Web consortium.

4.1.2 IBM food trust

Food trust is a traceability initiative by IBM. It is a blockchain-based system offering traceability and document management. The solution is made for the entire supply chain, from farmer to the consumer. Large corporations like Carrefour have already joined the system. Data are stored on the private IBM blockchain. The costs of food trust was 100 dollar for organisations with revenue lower than 50 million (Dignan, 2018), but this has information has disappeared from their website now (IBM, n.d.-b). Users can specify which parts of their data they are willing to share, with whom and under which conditions. Unfortunately, minimal information could be found on the data processing rights of IBM. It is therefore expected that IBM has the right to process users data for insights. Initiatives, like Thank my farmer for tracing the origin of coffee beans, are built on the Food trust network (IBM, n.d.-a). Although food trust is built on a blockchain, the costs of Food trust are relatively low at 100 dollars for small organisations (Dignan, 2018). Additionally, IBM provides an estimator tool for estimation of the saved products when implementing Food trust, the tool indicates a positive business case for businesses to join the system.

Traceability - data: Satisfied, food trust is capable of creating traceability

Traceability - linked: Satisfied, the response time of 2.2 seconds to find the origin farmer of products indicates that product links are established.

Traceability - query: Satisfied, Because the blockchain provides a location where queries can be sent to, to gather the traceability data required.

Traceability - backup: Satisfied, the blockchain is expected to provide the backup traceability

Access control: Satisfied but doubtful. Data on the IBM blockchain is stored encrypted, the user remains in control of their data and can define who has access. However, it is unsure if IBM sells the data to third parties (see supplier reputation)

Value proposition - easy to use: Satisfied, the application seems to be relatively easy to use (aelf, 2019).

Value proposition - compelling feature: Not satisfied, Food trust does not have a compelling feature. One of the benefits for adopters of Food trust

is the ability to validate the certification of products, this is likely to save a lot of time (aelf, 2019). However, this relies on the entire upstream supply chain adopting food trust.

Value proposition - data reuse: Satisfied, food trust offers a separate module for sale to upload import-documents and reuse data.

Trusted software supplier - governance: Satisfied, the governance of the system lies at IBM as the owner of the software.

Trusted software supplier - reliable: Satisfied, the system is expected to be reliable, based on the reputation of IBM to build reliable software. Additionally, the use of the blockchain should ensure a reliable system which is always online. This system generates a lock-in for the IBM-food trust. Because the used blockchain is privately owned by IBM.

Trusted software supplier - supplier reputation: Satisfied but doubtful, IBM is a somewhat neutral software supplier. Unfortunately, it is unsure whether IBM has access to the data of the supply chain to sell or not. The website says the data owner is in control of who can see the data, but it does not say anything about IBM using the data. Moreover, user agreements do not provide the required information to answer this question. Moreover, users do not pay for placing data on the blockchain, third-party apps that use the data do (Morris, 2019). This implies that IBM can access and sell the data of the users. Food trust is expected to generate a lock-in for their system. IBM has been around for a long time and is therefore expected to be for the foreseeable future.

4.1.3 TE-FOOD

According to Medium, "TE-FOOD is the most popular farm-to-table food traceability solution" (TE-FOOD, 2019). It is currently in use by at least 6000 business customers (TE-FOOD, 2019). TE-FOOD is a blockchain-based traceability system. Additionally, it has an API that accepts excel files and GS1 EPCIS formats for legacy systems. For organisations with minimal digital capabilities, TE-FOOD offers a mobile application to capture the required traceability data. For consumers, TE-FOOD offers a free mobile app to present food history information. (TE-FOOD, n.d.)

Traceability - data: Satisfied, the application provides traceability

Traceability - linked: Satisfied, links between products are established.

Traceability - query: Satisfied, because TE-FOOD is blockchain based, every actor has a location to send queries to that span the supply chain.

Traceability - backup: Satisfied, the blockchain ensures the backup traceability.

Access control: Not satisfied, the application provides minimal access control. The permissioned blockchain requires permission to write data to, but the transparency implies open data for other users and consumers.

Value proposition - easy to use: Satisfied, TE-FOOD seems to be easy to use (TE-FOOD, 2018). Especially with the use of their mobile application, traceability data can be entered on demand.

Value proposition - compelling feature: Not satisfied, there is no compelling feature because the application does not provide value when only one actor joins the system.

Value proposition - data reuse: Satisfied, TE-FOOD offers the opportunity to reuse data from excel and their mobile application.

Trusted software supplier - governance: Satisfied, TE-FOOD has its own governance.

Trusted software supplier - reliable: Neutral, Judging by their lowering amount of daily volume on their cryptocurrency, users seem to lose interest in TE-FOOD which will negatively affect the chance of survival. The use of the blockchain should ensure a reliable system which is always online.

Trusted software supplier - supplier reputation: Satisfied, TE-FOOD appears to be a neutral software supplier. Because their goal is to realise food traceability, this may work well for the agri-food supply chain. TE-FOOD does introduce a lock-in for their systems, but it allows freedom in the usage of third party apps for generating data to be uploaded to their blockchain.

4.1.4 SeafoodIQ

SeafoodIQ is a traceability system meant for the seafood industry. The system includes RFID tags in pallets and boxes to trace the products as well as the temperature during transportation and storage. The tags are also used in automatically tracking the movement once the tag moves through gates. In this way, SeafoodIQ appears as a complete system for both internal and external traceability. The data are stored on the blockchain. (SeafoodIQ, 2018a) SeafoodIQ is a project sponsored by the EU and expected to enter the market at the end of 2020 (*SeafoodTrace: Intelligent Traceability Platform enabling full transparency in the Seafood supply chain*, 2018).

Traceability - data: Satisfied, SeafoodIQ offers traceability.

Traceability - linked: Not satisfied, it does not appear to provide links between products. For the seafood industry this is most likely not necessary, as the product is a fish, and not/minimally used as an ingredient in subsequent products. The ability to query the data is present due to the use of a blockchain.

Traceability - query: Satisfied, similar to the previous two solutions, SeafoodIQ is built on blockchain, which acts as a location that actors can query.

Traceability - backup: Satisfied, the blockchain provides the backup traceability of the products.

Access control Not satisfied, The website of SeafodIQ does not provide information on the access control or security of the system other than "Enterprise-level security". However, it is expected that some form of access control is in place. The promo video indicates that restaurants can see all the information on the food they received, indicating open access to SeafoodIQ users (SeafoodIQ, 2018b).

Value proposition - easy to use Satisfied, the system appears to be very easy to use. The combination of RFID tags and automatic scanning gates saves manual work in keeping a record of product locations and movement. The temperature-sensitive RFID tags help in estimating shelf life and providing proof of shelf life (SeafoodIQ, 2018b).

Value proposition - compelling feature Satisfied, one of the compelling features would be the near-automatic internal and external traceability. Even when no other players would join the ecosystem, SeafoodIQ creates internal traceability and insights into the local production process.

Value proposition - data reuse Satisfied, although SeafoodIQ offers a complete system, acquiring all the hardware may require a large investment. Additionally, the temperature-sensitive RFID tags may create a lot of costs over a temperature measurement of the cool storage facility. However, the application automatically reuses all the data generated by separate RFID tags.

Trusted software supplier - governance: Satisfied, SeafoodIQ has their own governance.

Trusted software supplier - reliable: Satisfied, because they are founded in 2015, it is unsure for how long they will exist. The use of the blockchain should ensure a reliable system which is always online. However, the blockchain also generates a lock-in to their system.

Trusted software supplier - supplier reputation: Satisfied, SeafoodIQ is a for-profit organisation, making it not entirely neutral. Their system would create a lock-in when adopted.

4.1.5 Traces

TRAde Control and Expert System (TRACES) is a traceability system of the EU. It is meant for the documentation of imported food inside the EU and between member states. Traces is in use by all customs offices in the EU (*TRACES: TRAde Control and Expert System*, n.d.).

Traceability - data: Satisfied, the system captures traceability data by enabling actors to upload their relevant data.

Traceability - linked: Satisfied, the promotion video implies that there is a link present between the products at different owners by being able to directly determine the origins and destinations.

Traceability - query: Neutral, Traces offers a search function that has underlying query functions. However, it does not has a field for specifying queries.

Traceability - backup: Satisfied, most likely there is a backup of the data entered into the system somewhere.

Access control: Not satisfied, Traces does not provide the ability to determine who has access to the data. Additionally, competent authorities may always access the data in traces, outside the control of the actors.

Value proposition - easy to use: Neutral, Traces is available in 35 languages, this can create ease of use, especially in trade spanning country borders. However, the number of certificates and steps indicate a system which costs a lot of effort to use.

Value proposition - compelling feature: Satisfied, Traces does not provide a compelling feature. However, it is free to use, which is quite compelling. Additionally, the use of traces is obligated for some sectors, removing the need for a compelling feature to gain sufficient adoption in those sectors.

Value proposition - data reuse: Satisfied, Traces provides a service interface that can be used by third-party applications to use traces.

Trusted software supplier - governance: Satisfied, Traces is an initiative by the EC, and therefore the EC has the governance.

Trusted software supplier - reliable: Satisfied, Traces is available 24/7 and used by a large number of users, the system is expected to be reliable. Traces is mandatory in some sectors. Therefore it has a lock-in.

Trusted software supplier - supplier reputation: Neutral, Traces provides a helpdesk, which can contribute to a good reputation. However, as the EC is a government agency it is not perceived as neutral by the actors (Minnens et al., 2018).

4.1.6 Gap analysis

In table 13, an overview of some traceability systems can be seen. None of the systems fully satisfies the requirements. As can be seen, only IBM food trust satisfies the access control requirement. However, the policies hidden in the use of IBM food trust make both the supplier reputation and access control

Requirement	Linked pedigrees	IBM food trust	TE-FOOD	SeafoodIQ	Traces
Traceability - data	+	+	+	+	+
Traceability - linked	+	+	+	-	+
Traceability - query	+	+	+	+	+/-
Traceability - backup	+	+	+	+	+
Access control	-	+ and -	-	-	-
Value proposition - easy to use	+	+	+	+	+/-
Value proposition - compelling feature	-	-	-	+	+
Value proposition - data reuse	+	+	+	+	+
Trusted software supplier - governance	-	+	+	+	+
Trusted software supplier - reliable	+	+	+/-	+	+
Trusted software supplier - supplier reputation	+/-	+ and -	+	+	+/-

Table 13: Summary of the fulfilment of requirements for some of the commercially available traceability systems. systems are scored on the following scale: not satisfied or not possible (-), neutral (+/-) and satisfied (+).

doubtful. Moreover, none of the traceability systems has a compelling feature. Traces does not truly satisfy the requirement for a compelling feature. However, by making it available for free, actors may be compelled to use Traces. Next to that, the obligation to use Traces in some sectors removes any choice and thereby the need for a compelling feature. All systems have some a lock-in to some extend. Therefore two gaps are present in the current systems: access control and the compelling feature.

4.2 Approaches for traceability systems

4.2.1 Solid

Social Linked Data(Solid) is a set of standards and best practises which creates a decentralised platform for social web applications. Solid puts users back in control of their data. In contrast to many concurrent applications, Solid separates data and application. Every actor stores their data in a Personal Online Datastore(POD) at a service provider of their choosing (POD server). Applications run as web applications on client-side in a browser or as mobile applications (Mansour et al., 2016). Applications read and write data to and from the pods, and multiple applications can use the same data. By decoupling the data from the application, users can easily switch between similar applications without recreating their data as in current systems. Users are can switch between POD servers. For authentication, users register at an identity provider. This can also be the POD server. PODs can include both structured (RDF) and unstructured data of any type for videos and images. At this moment there seems no restriction on the vocabulary used, creating the possibility for multiple app makers to use or create different vocabularies, and thus making the true interoperability

between applications questionable.[1]

Requirements checklist

Traceability - data: Satisfied, Solid in itself does not create traceability. At this moment, Solid is designed as a social platform. A traceability system can be seen as social media for organisations where every actor posts their daily sells instead of daily pictures of normal social media. And a friend request can be seen as a data request. Therefore applications on the Solid platform could potentially create traceability.

Traceability - linked: Satisfied, the use of RDF naturally lends itself for the creation of links between products, even if the data are separated.

Traceability - query: Satisfied, The Solid PODs have been designed in a way to mimic a triple store endpoint. However, it does not yet possess a query engine that can be used to create SPARQL queries. But components can be written that execute as if they were queries.

Traceability - backup: Neutral, the backup traceability goes against the goal of Solid to prevent separate data silos, but this could be solved by a separate trusted "backup" or ILPS POD to which all users send their URIs.

Access control: Satisfied, the access control of Solid is good enough for the agri-food supply chain context. The UNIX like access features for read, write, append and control for each document is fine-grained and creates a lot of possibilities. However, depending on the application, the access control may be too complicated and cost a lot of time in configurations. Data can be made public and private and only accessible by pre-defined actors.

Value proposition - compelling feature: Neutral, there is no direct compelling feature in Solid. But Solid creates possibilities for data reuse between applications. This can create the possibility to use the data generated in the traceability system for other purposes and together create value, even when part of the supply chain is missing.

Value proposition - easy to use: Satisfied, because Solid primarily provides a backbone to build applications on, it is hard to judge on the ease of use for end users, as this is application dependent. But the reuse of data between applications can prevent double manual data entry and create ease of use there. Additionally, the setup of a Solid environment should not require a lot of technical knowledge. This is because data storage is done by the POD servers, and applications run on browsers. Another feature that creates ease of use is the combination of RDF and WebID,

[1] More information on Solid can be found on https://solidproject.org/

hiding the difficulties of authentication and routing of nodes from the users of the ecosystem.

Value proposition - data reuse: Satisfied, Solid claims that it is easy to build applications on their platform. If this is true, applications may become cheaper because it saves development time. The storage of PODs will present a cost factor for the end users, but this is also the case for contemporary storage solutions. Another possibility to reduce costs is to grant the POD server access to the stored data for analytics to gain free POD storage. Another possible way Solid could provide cost savings is trough a possible better organisational fit. Because in Solid, multiple applications can work on the same data, new applications may become available without the need to gather additional data with the new application. This can prevent part of the costs in adapting business processes to new applications.

Trusted software supplier - supplier reputation: Satisfied, the Solid architecture solves vendor lock-in problems by creating the possibility to switch between applications while keeping and reusing your data in a competitors application. However, the choice for Solid locks actors into the solid environment.

Trusted software supplier - reliable system: Satisfied, because solid is a set of open standards and not a direct application, it does not provide insurances for a reliable system. Similarly, the reliability of internet connections does not stem from W3C but internet providers. However, Solid indirectly enforces reliable systems. The created environment where users rapidly switch between applications punishes unreliable systems by a rapid loss of users and thereby enforces reliable applications.

Trusted software supplier - governance: Satisfied, because Solid relies on open standards from the W3C, it has a governance structure present.

4.2.2 IDS

Another approach initiative is the International Data Spaces (IDS). In some aspects it has similar features but is has different goals. Where Solid is designed for social data, IDS is designed to create interoperability between organisations in an data sovereign way. IDS is built on three assumptions: "Data is a strategic and economically valuable resource", companies will have to share data more and more to stay competitive and companies only share data when data sovereignty is guaranteed. Therefore IDS grants data owners control over their data and the possibility to sell data with usage agreements between two parties. (Otto, Steinbuß, Teuscher, & Lohmann, 2019)

In IDS every actor connects to the IDS environment trough their "connector". The connectors send and receive data from the applications of users. IDS assumes that software companies will build connectors for interested companies.

Each actor only has to create a connection between their software and their connector. Each connector registers itself at a broker with a list of their offered or requested data. Actors use the broker to find other actors that offer the data they need. The connectors establish a protocol for data exchange. Data exchange is registered at the clearinghouse to resolve conflicts and create the possibility to sell data between actors. An identity provider is used to prove everybody is who they say they are. Service providers are available to host the applications and connectors. Because IDS does not have a mandatory protocol, these have to be established for each integration project. This may increase the costs of IDS projects over projects that have a protocol established. On the other hand, IDS creates the possibility to sell data, possibly lowering the costs of IDS projects.[2]

Requirement checklist

Traceability - data: Satisfied, IDS should be capable of creating traceability. However, there are no restrictions on the data used. This can both be an advantage and disadvantage, as every actor can stick to their applications and data models.

Traceability - linked: Satisfied, When implemented correctly, links between products should be possible.

Traceability - query: Neutral, each actor has to search for a data provider, and can then query them through the connector component. Unfortunately, this requires the actor to find the data provider before a query can be made.

Traceability - backup: Satisfied, Because all data transactions are stored at the clearinghouse, this actor may also play a role in the backup traceability.

Access control: Satisfied, Access control happens through the connector. The connector checks if a data requester is certified. When sharing data, usage policies are included containing restrictions like the non-persistence or re-sharing of data.

Value proposition - compelling feature: Satisfied, the most prominent compelling feature of IDS is the possibility to sell data under predefined policies creating the possibility for new business models as well as the possibility to materialise the value of their data without the need for the entire supply chain to join.

Value proposition - easy to use: Not satisfied, at this point, it is difficult to judge if IDS is easy to use. The broker provides the opportunity to find data providers which may improve the ease of use. On the other hand, the number of roles can be perceived as complex. Additionally, making the connector to function with the environment may cost a lot of

[2]More information can be found on https://www.internationaldataspaces.org/

work. Finally, IDS does not provide protocols for the data exchange and this should be negotiated between the connectors. While, for integration projects, agreeing on a protocol is one of the difficult steps.

Value proposition - data reuse: Satisfied, IDS does not place any limits on current applications. Therefore it does not enforce large business transitions to adopt new applications. Old applications can be reused once a connector is in place. Additionally, the connector can enable data reuse.

Trusted software supplier - governance: Satisfied, The development of IDS is done by the International data spaces association. With industry and research organisations as members. Because the IDS association contains a wide variety of members it is expected to be relatively neutral.

Trusted software supplier - reliable: Satisfied, The IDS environment is expected to be in use for a long time since the governing association contains a lot of members which exist for a long time. The system is vulnerable when an actor is not available to become unusable for their partners. By choosing for IDS, a lock-in to the infrastructure happens. Moreover, it could harden the switch between applications, as a new IDS connector is required to connect to the infrastructure.

Trusted software supplier - supplier reputation: Satisfied, Applications on the IDS environment depend on the manufacturer. When choosing for IDS, a lock-in for this environment is present. However, since the connector is the link to the IDS environment, individual applications can be kept and possibly interchanged.

4.2.3 Inter planetary file system

Inter planetary file system(IPFS) is a decentralised and distributed file storage system. Files are stored and made public for everybody to read. When a file is requested, the file is downloaded from multiple sources. IPFS stores files based on content rather than location-based like file systems and HTTP. When somebody is interested in a file, they download the file and make it available for other interested persons. In this way, every file is available as long as at least one person has it (Labs, n.d.). IPFS is expected to lower the costs hardware to provide enough bandwidth to handle requests for data, as part of this is taken over by other parties interested in the data.

Requirements checklist

Traceability - data: Satisfied, IPFS should be capable of storing and sharing traceability data. Each actor could publish their data on IPFS and others could copy it.

Traceability - linked: Satisfied, because the content is stored based on content rather than location, product links could be made directly into the files.

Traceability - query: Satisfied, IPFS can query across multiple data sources.

Traceability - backup: Satisfied, because multiple people have the data the backup traceability is almost guaranteed. However, when nobody publishes a file, it disappears.

Access control: Not satisfied, the current IPFS does not provide the ability to share files with selected users for sensitive files (Steichen, Fiz, Norvill, Shbair, & State, 2018).

Value proposition - easy to use: Not satisfied, similar to the previous solutions, it is hard to judge the ease of use. The content-based storage instead of location-based storage may be difficult for end users, as most concurrent systems work location-based. However, a good interface can prevent this.

Value proposition - compelling feature: Not satisfied, Although IPFS creates increased bandwidth for file downloads which can be compelling for certain industries. IPFS does not bring a compelling feature.

Value proposition - data reuse: Satisfied, IPFS has the possibility to reuse available data.

Trusted software supplier - governance: Satisfied, IPFS is an open-source project by Protocol Labs.

Trusted software supplier - reliable: Neutral, because IPFS is relatively new, Adopting IPFS is expected to bring risks, as it is unsure whether it will still exist in 10 years. Additionally, the reliance on people having an interest in specific files may pose a threat to the system, but also a way to guarantee 100% uptime. If IPFS is used, it is likely to create a lock-in for their environment, because of their unique architecture.

Trusted software supplier - supplier reputation: Neutral, IPFS is an environment rather than an individual application. Therefore there is supplier reputation of the application is yet unknown. IPFS has extensive documentation and tutorials that help programmers in implementation.

4.2.4 Blockchains

Blockchains store data in a tamper-proof, decentralised manner. Data is stored in blocks. Each time new data are added, a new block is created and added to the chain. Each block contains a hash of their previous block and this creates the security of the blockchains. Because each block contains the hash of the previous block, old blocks cannot be changed without changing all the future blocks. This means that data on the blockchain are immutable. In a blockchain, the majority of the network have to agree on the validity of the next block. Blockchains have multiple ways of generating new blocks, checking their validity and prevent tempering like the proof of work. There are many

versions of blockchains with different technologies behind in, but their basic principles as described above are similar. Blockchains can be open, like bitcoin, and permissioned like Ethereum. The former is open for everybody read from and write to while the latter requires permission to do so. The blockchain is more expensive than traditional data storage. The costs of storing data on the blockchain are about 2000 to 8000 times more expensive than on a cloud-based database (Content blockchain, n.d.). Additionally, the business model of a blockchain is doubt-full. Because a blockchain is immutable, the data are on the blockchain forever. This, in contrast with the concurrent trending for service business models like the I(nfrastructure)aas, S(oftware)aas and P(latform)aas. In these models, everything relies on a monthly payment, while the blockchain would rely on a one-time payment for the storage of a block for eternity. This will eventually realise a collapse of the blockchain because the new blocks will not generate enough profit to sustain all the previous blocks, requiring higher block prices or removal of old data. Some applications exist with a monthly fee, running on a permissioned blockchain of a third party (IBM, n.d.-b).

Requirement checklist

Traceability - data: Satisfied, blockchains are capable of creating traceability.

Traceability - linked: Satisfied, linking between the data records is possible.

Traceability - query: Satisfied, queries can be sent to a node of a blockchain.

Traceability - backup: Satisfied, because many nodes in the blockchain network contain the data, backup traceability is ensured.

Access control Neutral , access control depends on the blockchain chosen. Some are open to read for every node, while others require permission to join the network for reading and writing data. Some blockchains offer the ability to place smart contracts, which are pieces of code, placed on the blockchain. These can be used to create access control policies on the data in the blockchain. Unfortunately, since the blockchain is readable to the users, so is the smart contract, laying bare possible implementation mistakes in the access control policy. Therefore, this requirement is deemed unfulfilled.

Value proposition - easy to use: Neutral, similar to the previous approaches the blockchain does not provide direct applications and ease of use cannot be judged at this point.

Value proposition - compelling feature: Neutral, one of the compelling features of using a blockchain is the current hype attached to it (Panetta, 2019). A blockchain-based traceability system probably has a good selling point. However, the blockchain in itself does not provide value for actors when others do not join.

Value proposition - data reuse: Satisfied, blockchains enable data reuse. Because all data can be stored on the chain, and accessed by all the nodes, data can be accessed and written by a multitude of applications of a single actor.

Trusted software supplier - governance: Neutral, blockchains are developed by a wide range of actors, being individuals and organisation. Therefore blockchains can sometimes be built by neutral organisations and sometimes by commercial organisations like IBM.

Trusted software supplier - reliable: Neutral, when choosing for a blockchain, a lock-in is generated for the blockchain technology. It could be possible that similar blockchains could be changed with minimal effort, but that would require the entire supply chain to switch. Another possibility would be when multiple applications built on the same blockchain. This limits the lock-in for one application but creates a lock-in for that blockchain. Because the blockchains are relatively new, it cannot be guaranteed to be present in the future. However, the hype increases the chance that blockchains are present for the foreseeable future. But care should be taken that when the nodes lose interest in a certain blockchain, that blockchain becomes insecure and data may be lost. However, as long as enough nodes have interest in the chain, it is always available.

Trusted software supplier - supplier reputation: Neutral, because blockchains come in a wide variety of versions. The supplier reputation should be judged for each blockchain.

4.2.5 Gap analysis

In table 14 the requirements checklists have been placed in table format. Unfortunately, none of the systems completely satisfy all the requirements. IPFS is not suited to built a traceability system on, as it does not comply with too many requirements. The other three are possible. However, every approach has its downside. Solid's design challenges the implementation of a backup. Next to that, it does not provide a compelling feature. IDS checks all the boxes but has the downside of creating a complicated ecosystem. Blockchains are very strong in the functional points of traceability(data, linked, query and backup). However, it brings a lot of new issues that have to be overcome before it can be used. All approaches generate a lock-in to some extent. In conclusion, Solid seems the most appropriate choice for an approach because it satisfies most requirements and the challenges it has seem manageable.

4.3 Ontology based access control

As part of this research, the possibilities that Ontology based access control(OBAC) (Brewster, Nouwt, Raaijmakers, & Verhoosel, 2019) brings to the LP architecture this is analysed. First OBAC is introduced. Then the advantages and disadvantages are shown.

Requirement	Solid	IDS	IPFS	Blockchains
Traceability - data	+	+	+	+
Traceability - linked	+	+	+	+
Traceability - query	+	+/-	+	+
Traceability - backup	+/-	+	+	+
Access control	+	+	-	+/-
Value proposition - compelling feature	+/-	+	-	+/-
Value proposition - easy to use	+	-	-	+/-
Value proposition - data reuse	+	+	+	+
Trusted software supplier - supplier reputation	+	+	+/-	+/-
Trusted software supplier - reliable system	+	+	+/-	+/-
Trusted software supplier - governance	+	+	+	+/-

Table 14: Requirement satisfaction of state of the art approaches. The requirements are scored on a three point scale: not satisfied or not possible (-) or neutral (+/-) or satisfied (+)

OBAC is an access control framework designed to grant access to data "in settings where data access is demanded but restricted" (Brewster et al., 2019). The idea of OBAC is a generic one and abstracts from implementation details like APIs and resources. Therefore OBAC can potentially be used in multiple cases, regardless of the underlying architecture. To achieve this, OBAC has the following assumptions to function (Brewster et al., 2019):

- "Metadata needs to be assigned to raw object data prior to access."
- "The metadata scheme adheres to an ontology: it is hierarchically structured, with meaningful (interpretable, semantic) relations between nodes (concepts) and reflects domain knowledge."
- "Access to the object data occurs through the metadata, with the possibility of defining access for a given person or role to specific layers (strata) in the metadata."
- "Access to object data and re-usability of (meta)data is determined by referring to the structure of the metadata graph, the contents of the nodes, or both."

This is interpreted as data must be assigned metadata; the data are structured as an ontology; A data requests happen by the relations in the ontology; Access policies are created by defining the structure a person or role has access to; access is granted or rejected based on the pattern of the request and value of the data records.

In their proof of concept, OBAC is built as an extension of the XACML architecture and a SPARQL endpoint. It utilises the policy enforcement point to catch the request for data. In OBAC each data request is (part of) the meta data graph. In the policy decision point, the request is evaluated. To allow

or reject access, the request graph is compared to the policy graph. Only the matching part of the graphs is returned to the user.

Another Access control framework is Web Access Control(WAC) used in Solid. WAC has a similar structure as access control on a UNIX file system. Each container or file is accompanied by an access control file. In this file, the access modes, the affected files and the users are defined. In comparison to OBAC, WAC has a higher granularity. Where OBAC creates one role for multiple resources, WAC creates multiple roles for one resource. Another difference is that OBAC relies on a meta data description for access policies, while WAC explicitly writes all the access modes for a resource.

An extension on WAC is pattern based access control(PBAC) (Werbrouck et al., 2020). PBAC and OBAC both utilise the idea of matching a pattern to determine access to the data. In OBAC the focus lies on patterns in the data and constant roles or users are assumed. In contrast, PBAC focuses on the patterns available in roles of users to determine access to a file rather than individual triples. In future work, PBAC intents to extend the methods to use ontologies to describe the user as well, possibly combining both approaches.

One of the advantages of OBAC is the possibility to create a single access policy for multiple instances of resources. This allows for easier maintenance of access policies because less policies are required. Moreover, access based on the graph combined with values in the graph, creates the opportunity to grant access to a single triple in a triple store. This contrasts with contemporary systems where all or nothing access is granted to files. Next to that, the potential use of SPARQL allows easier development of systems that use cross-server queries (Mansour et al., 2016). However, the use of OBAC also brings some disadvantages. First, applications are bound to have a SPARQL like endpoint, where data is requested based on the meta data graph of the used ontology. Moreover, because OBAC requires the use of an ontology, it prevents possible unstructured data to be found.

In the context of the agri-food supply chain, OBAC brings the advantage of sharing a specific part of the data. By limiting the amount of shared data with an actor, the potentially sensitive parts of the data can be screened off. This could lower the perceived risk of sharing data in a traceability system, and thereby increase the adoption chances. Another advantage of sharing only the required amount is that it lowers the amount of bandwidth required to retrieve the data. Furthermore, it allows for a more efficient search in the data by the data requester. This is because the data requester only receives the data they demand, not the entire file where he searches for the correct part of the data. Finally, the agri-food supply chain is expected to fill many of the same forms. With the use of OBAC, the access policies for data in these forms only has to be defined once. However, OBAC also has some disadvantages when used in the agri-food supply chain. First, all the data must be structured and in line with an ontology. However, as most actors still work paper based this is not possible. For the ones that have digital systems, there are a lot of different standards used (section 2). This means that for OBAC to work in an interoperable environment, actors must use either the same ontology or a

component translating the different standards to an ontology used at another actor. Moreover, OBAC may experience the role explosion observed in role based access control frameworks, as it relies on the definition of access policies per actor or role (Elliott & Knight, 2010). Next to the amount of access policies is the potential difficulty in defining the access policies. Correctly defining the access policies requires the end user to sufficiently understand the used ontology. As an ontology can be perceived as complex, this could conflict in with the adoption of systems using OBAC.

5 Architecture

The next part of the treatment design is the design of new treatments. For the development of the new treatment, The open group architecture framework (TOGAF) was used because design science focuses on a meta-level rather than detailed guidance in the treatment design phase. TOGAF is an architecture framework for enterprises. TOGAF defines 9 sequential phases. The phases start with a preliminary phase which focuses on defining project constraints[3]. Phase A focusses on the stakeholders, their concerns and objectives, requirements, and constraints[4]. Therefore, the activities in phase A are described in section 2 and 3. Phase B[5],C[6] and D[7] focus on the creation of the architecture. This section is structured according to these three phases. The goal of Phase B(section 5.1) is to describe the strategy and business environment. In phase B, a baseline architecture(current/as-is state) and target architecture(to-be state) are made. The target architecture should realise the stakeholder goals. The baseline architecture and target architecture are used to identify gaps between the organisational states. The gaps are used in phase C (section 5.2) to identify the information systems components required to realise the stakeholder goals. Phase C is split into the data and application architecture. In Phase D (5.3) the technology architecture is created, which realises the architecture components identified in phase C. This section stops at phase D because phase E and later phases focus on the implementation and transition road map.

Viewpoints in this section are made in the Archimate modelling language. This is the modelling language developed by the open group that complements TOGAF[8]. The Archimate modelling language relies on colours for the level of a concept(e.g. yellow for the business layer, purple for the motivation layer, blue for the application layer, and green for the technology layer) and concepts within each layer(e.g. goals, requirements, and stakeholders in the motivation layer)

5.1 Business architecture

5.1.1 Baseline description

Before a baseline description was made, that fits all the actors in the supply chain, a model of the ready-made lasagne supply chain was made. Most value chains are split in multiple actors to create parts of the end products. This is graphically illustrated in figure 3a. It illustrates the value creation process of ready-made lasagne. Figure 3a shows the business actors involved in the creation of ready-made lasagne on the right and left. Moreover, each actor realises part of the value chain, and does not realise value on its own. The sub-parts of

[3]https://pubs.opengroup.org/architecture/togaf8-doc/arch/chap04.html
[4]https://pubs.opengroup.org/architecture/togaf8-doc/arch/chap05.html
[5]https://pubs.opengroup.org/architecture/togaf8-doc/arch/chap06.html
[6]https://pubs.opengroup.org/architecture/togaf8-doc/arch/chap07.html
[7]https://pubs.opengroup.org/architecture/togaf8-doc/arch/chap10.html
[8]https://pubs.opengroup.org/architecture/archimate3-doc/

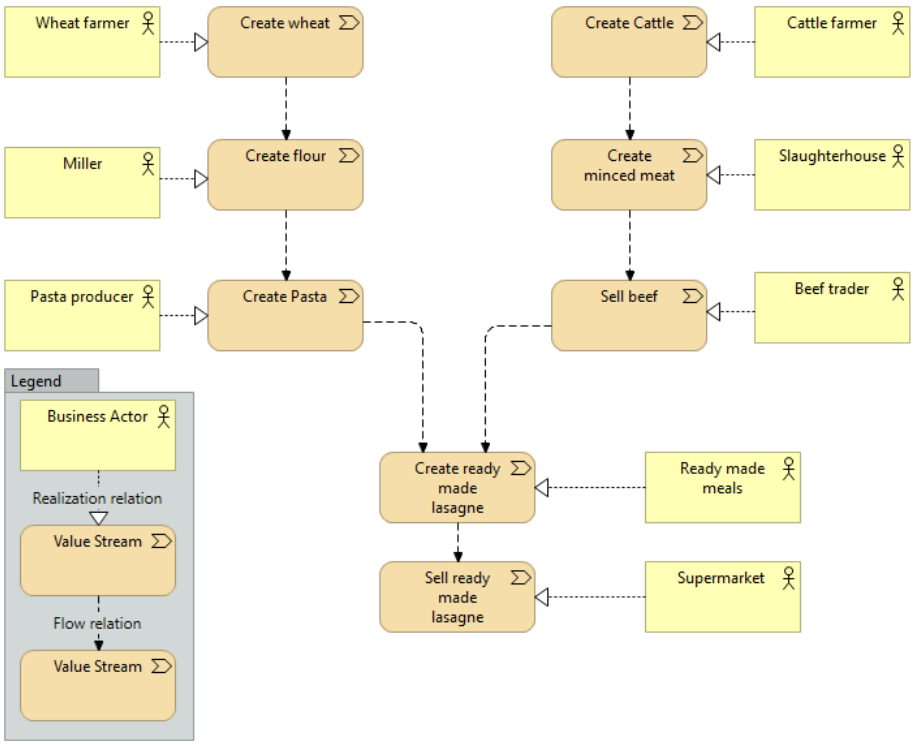
ready-made lasagne are depicted next to the actors. Each actor creates part of the ready-made lasagne and sells it to the next person. The primary reason for being part of the agri-food supply chain is the production of food for society. A food safety incident at a single actor hurts the value for the entire chain.

Figure 4a shows a generalised value stream of a single actor in figure 3. A generalised version is required to create a baseline architecture that holds for most actors. As their primary reason for being, each organisation is producing some part of a final food product. The production of a partial food product is represented by the *food production* value stream. Every agri-food supply chain actor must have a *food production* value stream, or they would not be part of the agri-food supply chain (aside from distributors and retailers). Three business processes should be present to realise the *food production*: *Buy ingredients*, *produce (partial) food product*, and *sell food product*. The details of food production were abstracted into *produce (partial) food product* because every actor does this different. The *register ingredients* and *register sold product* are present in every actor because they are required by law for traceability. These two processes realise the *traceability data* value stream. To support the food production, an *inventory management* process is in place. For a single farmer, this means feed for the cattle is in stock, while it can be realised by an ERP system or department in larger organisations. Moreover, large organisations may have additional business processes such as customer service. But these are not guaranteed to be present in every actor.

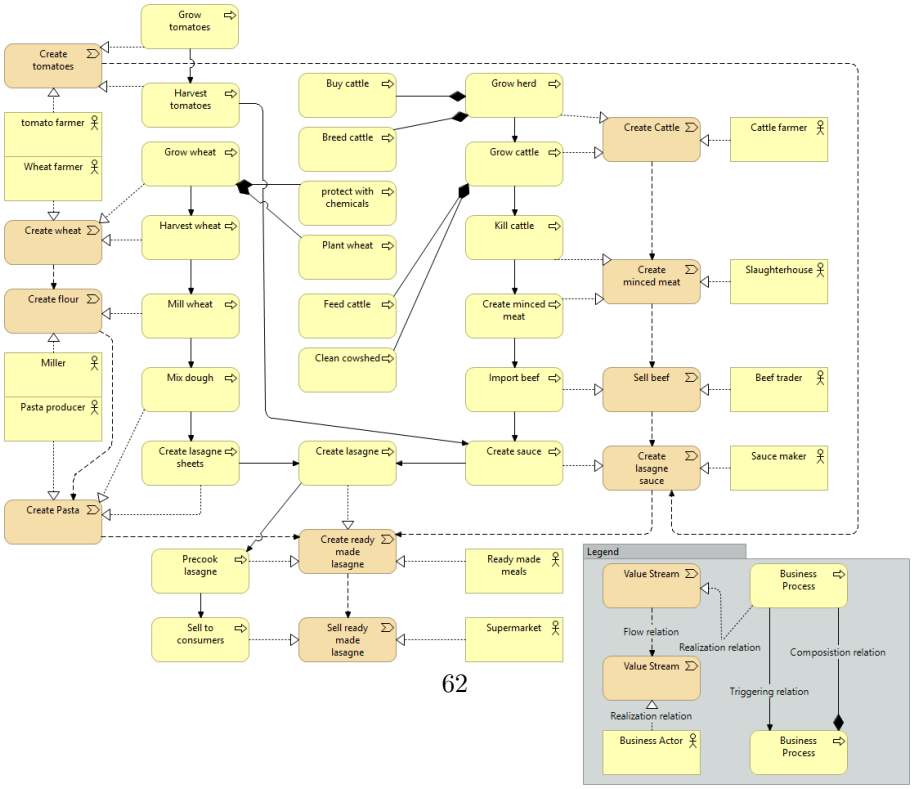
5.1.2 Target description

To argue for changes to the baseline description towards the target description, a motivation viewpoint is made. Figure 5 shows the motivation view. The motivation viewpoint links the requirements and concerns to the goals. This figure shows why stakeholders should transition to a target architecture. Figure 5, depicts the agri-food supply chain as stakeholder on the top. Because food safety incidents hurt the chain in their value creation, their goal is to *increase food safety*. This goal can be broken down into two sub-goals: having a *better response time*, and *early detection* of food safety incidents.

Better response time The target architecture should assist when a food safety incident happens. For example, during the horsemeat scandal, authorities found products contaminated with undeclared ingredients. In a target architecture, supermarkets should directly point to the producers of ready-made lasagne with the use of their traceability system. Assuming they have access to the data of the lasagne producers, the supermarkets could find the beef suppliers of the lasagne producer by following the traceability data. Combining the information of the lasagne producer and the contaminated products, it is now possible to determine which beef supplier sold contaminated beef. After identifying which beef supplier had contaminated beef, their sources can be investigated even further until the root cause is found. Because all the data are up-to-date, it should not take longer than a day to find all the

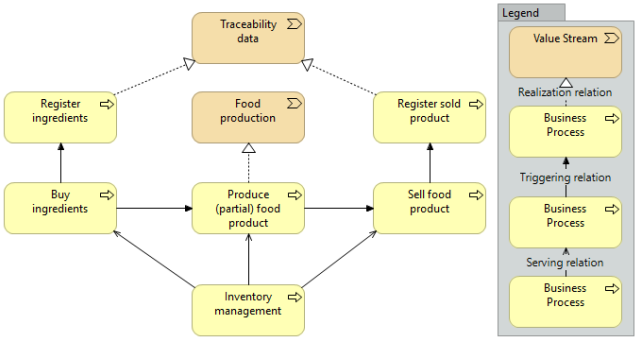


(a) Value stream of ready-made lasagne. Indicates two ingredients used in the creation of ready-made lasagne. Depicts the business actors and their addition to the value stream of ready-made lasagne.

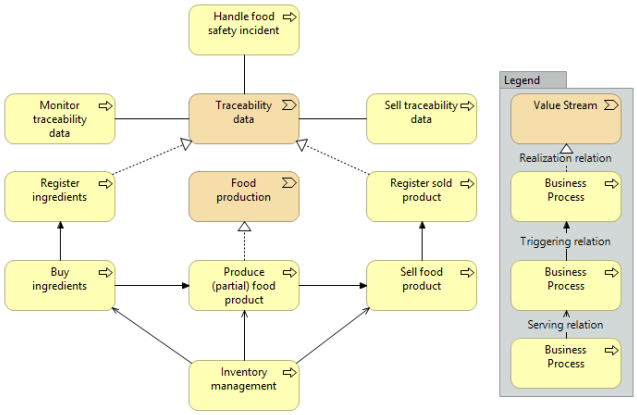


(b) Value stream of ready-made lasagne with more detail. Indicates the same value stream as 3a. The value stream is enriched with the business processes realising the value.

Figure 3: Models of the value stream of ready-made lasagne



(a) The baseline value creation process of agri-food supply chain actors. Their main value creation process (food production) is shown in the centre. As part of this process is the production, buying of ingredients, selling of the final product and creation of documentation.



(b) The target value creation process of agri-food supply chain actors. Their main value creation process (food production) is shown at the top and has not changed. As part of this process is the production, buying of ingredients, selling of the final product and creation of documentation. two new business processes are shown: sell traceability data and sell fair trade food.

Figure 4: Baseline and Target value creation process of an agri-food supply chain actor.

involved actors. The consequence of this time reduction is that the supply chain can perform mitigation actions within one week as opposed to the current five weeks. Then the actors can limit the amount of consumers that eat harmful products.

Early detection The target architecture should prevent food safety incidents. In the horsemeat scandal, part of the horsemeat came from a company called Draap which is horse spelt backwards in dutch. According to Romania, Draap correctly exported horsemeat while it was labelled as beef in France (Felicity, 2013b). The goal of the *early detection* is to discover these kinds of things. This may have been discovered when actors requested proof from their suppliers or when actors monitor their traceability data for red flags. In this way, the *early detection* can prevent harmful products from leaving the supply chain. Because hints to faults can be detected in the traceability data before a product reaches retailers.

Traceability realises the *better response time* and *early detection*. To handle according to the *traceability*, the data must be up-to-date. To realise up-to-date traceability data, data is *generated during production*. If the actors generate the data a month after producing a food product, a *better response time* and *Early detection* would become impossible. Moreover, the traceability system must satisfy the *requirements* (of section 3) to lower the barriers to adoption for actors. Because these goals can only be realised when a large proportion of the supply chain collaborates.

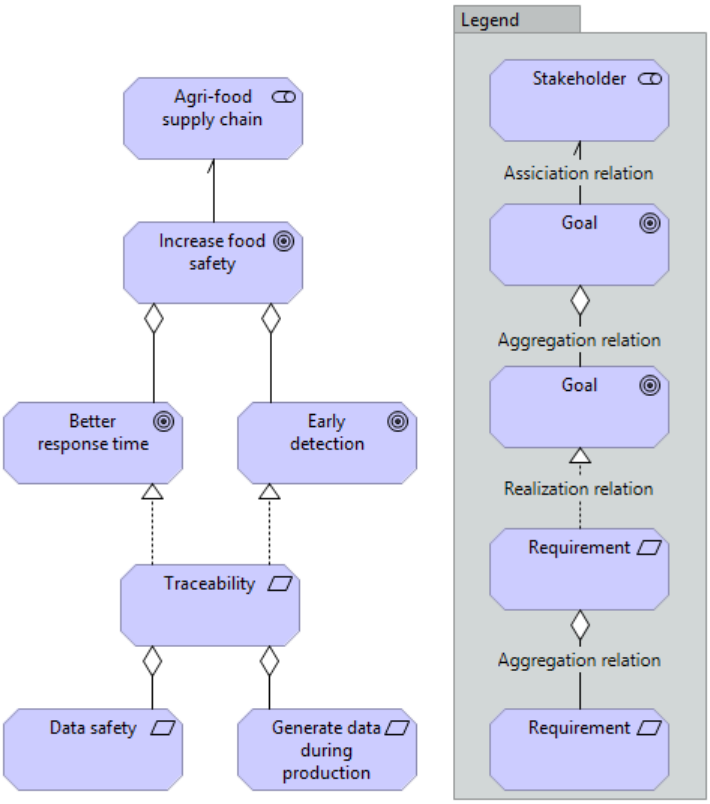


Figure 5: Motivation viewpoint. Shows the goals of the agri-food supply chain. The supply chain is placed on the top. Below them are the goals they want to achieve. On the bottom are the requirements that realise the business goals.

Where actors hesitated to share information with other actors in the past, the target architecture should allow organisations to share a very specific part of their information. This allows food safety incident discovery without sharing an entire dataset or confidential parts of it. During times without food safety incidents, the traceability data are kept up-to-date. This will ensure that data are available when it is necessary.

Figure 6 shows the realisation of the business goals of figure 5 in the target architecture. The highest goal of the supply chain actor is to *create value*. This is realised by the *food production*. The *food logistics* ensure enough resources are present for the *food production*. Together, the *food production* and *food logistics*, generate the *traceability data*. They do this by scanning the consignments that enter and leave the organisation. Additionally, the *food production* registers the ingredients they use. New capabilities use the *traceability data* to increase the food safety. The capability to *monitor traceability data* realises the *early detection*. Monitoring of the traceability data can be done by algorithms or

manually. The capability *handle food safety incidents* realises the *better response time*. The handle food safety incidents is a manual process where actors work together with food safety authorities to find the root cause. The *early detection* and *better response time* positively influence the *food safety*. Which positively influences the value creation.

figure 4b shows the result of the target description. It shows the new business processes enabled by a traceability system: the ability to *sell traceability data*, *monitor traceability data* and *handling food safety incidents*.

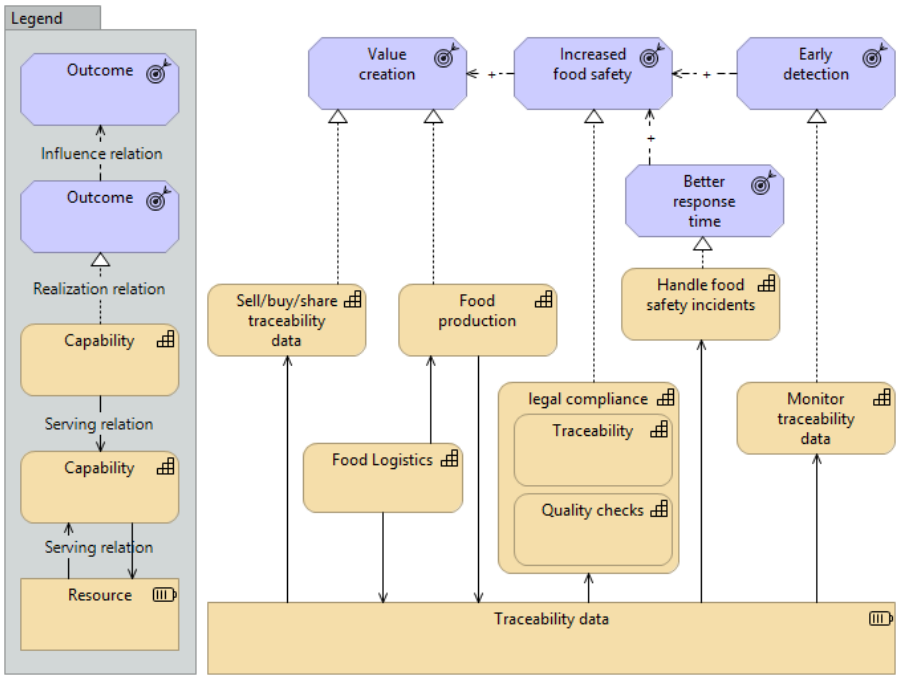


Figure 6: This figure shows the strategy viewpoint. These are possible strategies that businesses could use. Below them are the business capabilities that realise the strategies. The business capabilities rely on the traceability data. The traceability data are generated by food production and food logistics.

5.1.3 Gap analysis

In transitioning from the current way of working (figure 4a) to the new way of working (figure 4b) 4 gaps have been identified. These are listed below. Most business processes from the baseline architecture are conceptually the same. This has the advantage that current systems can be reused and minimal business remodelling is required. This should contribute to the organisational fit, as the actors can keep using their current systems and way of working.

1. **Monitor traceability data**

To realise the goal of early food safety incident detection, the actors should realise a business process to monitor the traceability data. This is one of the new business processes in the target architecture. Actors should setup new tasks and functions to look for strange behaviour in the traceability data. E.g. an organisation with the name meaning horse, selling cheap beef meat. This can be done manually or with automated algorithms. To bridge this gap, the actors would require data of partners to monitor for irregularities.

2. **Handle food safety incidents**
   During a food safety incident, actors should use their traceability data to handle accordingly. This is the second new business process. To bridge this gap, actors should define processes to perform mitigating actions as soon as possible. To do this, the actors use the traceability data to identify the affected products. Existing employees, management or a short time hired external expert can realise this process. Moreover, to handle food safety incidents, the actors must combine their traceability data to find the root cause of the problem.

3. **Digital traceability data gap**
   This gap is the new implementation of the value creation process *traceability data*. Although traceability data is present in both figure 4a and 4b, the implementation chances. This happens on the application level. In the baseline, capturing traceability data happens on paper while in the target, this is done digitally. The digital traceability data are required by the first two gaps. To generate the digital traceability data, the implementation of the *food production* and *logistics* change. In the baseline architecture, employees would fill in traceability data on paper. In the target, this happens on a digital device. Furthermore, employees register events at the moment they happen, so that data are up-to-date.

4. **Custom access control**
   This gap originates from the gaps to monitor traceability data and handle food safety incidents and concerns for the safety of traceability data. This creates a dilemma between sharing data and keeping it private. However, a middle way, where an actor can define who has access to specific parts of their data, can bridge this gap. When an actor can define access to its' data, they can share their data with the partners they trust. Moreover, they can sell this access, thereby realising the new business process to sell traceability data.

5.2 Information systems architecture

In TOGAF the information systems architecture is split into two parts: the application architecture and the data architecture. In the application architecture, the components are defined that bridge the gaps defined in the business

architecture. In the data architecture, the data are defined that are required by the application components to realise their function.

5.2.1 Application architecture

In figure 7, the high-level application architecture can be seen. The artefact is an implementation of the Linked pedigrees architecture on the Solid ecosystem. It is an update of the LP architecture because the access control was missing. The architecture shows 3 actors and their traceability application. Each application stores its data in a Solid POD. The ILPS is placed on the top of the figure.

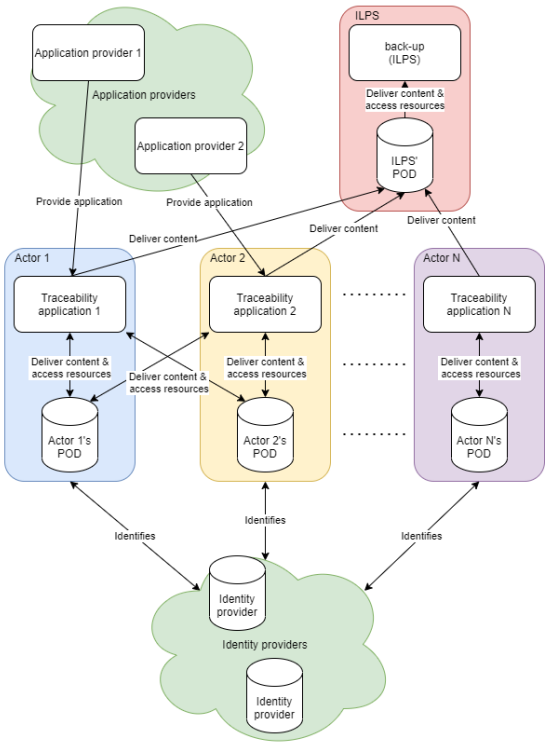


Figure 7: High level architecture. It shows three instances of the traceability system used by supply chain actors. The number of traceability systems equals the number of actors in a supply chain. It also shows the PODs of individual actors. It can be seen that the ILPS provides an overarching structure for backup traceability. The coloured boxes around a traceability system and POD, indicate one traceability system. The Identity providers enable secure connections between the traceability systems and the PODs

LP has a lot of potential and was therefore used as a starting point for an architecture. The choice for Solid as approach was based on the fit with LP and because it was one of the best choices of section 4. The architectures of LP and

Solid complement each other because they both use the same standards. Next to that, both architectures complement each other in the parts they are missing. LP provides an application that is required before Solid works. The downside of LP was that it lacked an infrastructure. Solid provides this. Additionally, the decentralised nature of both architectures makes them compatible. The final argument for Solid is the available libraries speeding up implementations.

The Solid ecosystem is the best solution for a traceability application. Solid brings more solutions and a better fit with LP in comparison with blockchains. Although blockchains are strong in satisfying the traceability requirements, they are less strong in the access control and the value proposition requirements. When comparing IDS with Solid, the same benefits can be realised. However, the IDS ecosystem involves more actors, which could decrease the ease of use. Moreover, Solid has a lot of libraries ready to be used while IDS does not. IPFS is not suited because it is not possible to create access control.

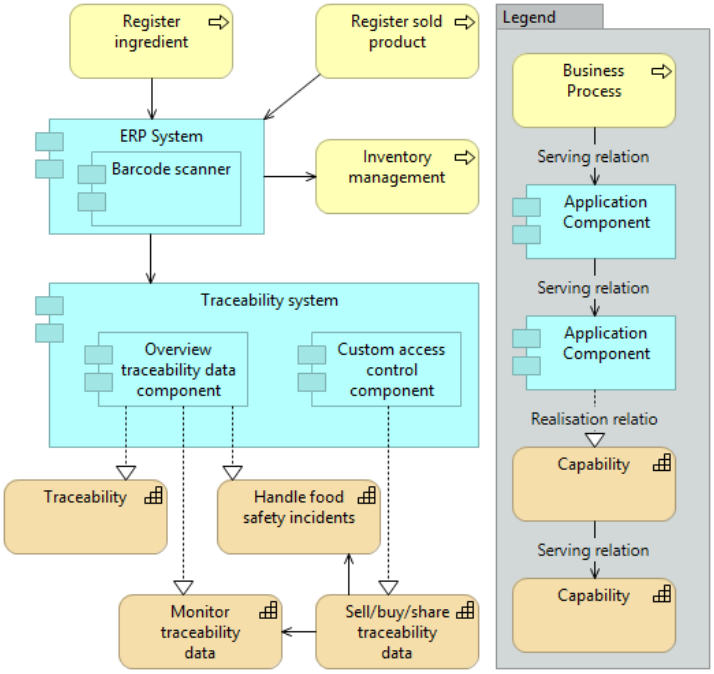


Figure 8: Process/Application realisation diagram. The top layer indicates the business processes that use application components.The ERP system serves the Traceability system with data. The traceability system realises the capabilities shown in the bottom.

Figure 8 shows how the business capabilities are realised by the application components. This is a process/application realisation diagram. Moreover, it shows which business processes use the applications. This creates the connection between figure 6 and 4b. The top layer of figure 8 depicts three of the business

processes (the same business processes as in figure 4b). The traceability system realises the capabilities of figure 6. The system realises the gaps identified in the business architecture in the following way:

- **Digital traceability** To realise the digital traceability, employees use a barcode scanner of an *ERP system* to register ingredients and sold products. The ERP system serves the *Traceability system* by generating the traceability data required. If the actor does not have an ERP system, employees enter all data into the traceability system during food production and logistics.
- **Handle food safety incidents** An overview screen that shows the traceability of produced products realises the handle food safety incidents capability. An overview screen allows actors to view their traceability data, and handle according to the information they see.
- **Monitor traceability data** Similarly, an overview screen, realises the monitor traceability data capability, Such a screen allows actors to look at data of their products and find irregularities.
- **Sell traceability data** A *custom access control component* realises the ability to sell traceability data. When actors can themselves define who has access to their data, they can make deals for this. Another option is to open their data for free to their partners in return for access to the partners' data.

figure 9 shows the traceability system in more detail. The goal of this diagram is to present the composition of the traceability system into parts that realise the functions the system must fulfil. The most important thing to notice is that Solid applications do not have a back-end. This means that all data transformations happen in the application, and the PODs only store data. Therefore, the application consists of two components: the *POD connection* and the *user interfaces*.

The POD connection reads and writes data to the PODs. The TripleDoc module realises the POD connection. The TripleDoc module implements the W3C standards used in the connection with PODs, more information on the Solid ecosystem can be found in the documentation[9]. Because the ILPS is a POD, the same standards apply. In the application, data flows always start in the user interfaces. The user interfaces use the POD connection to send requests for data to the PODs. The POD connection waits for the response of the PODs. If the user has access to the data, the connection sends the data to the interfaces where it can be used. The users can then change the data and save the data trough the POD connection.

The *user interfaces*(UI) of figure 9 are the access points for the end users. These UIs realise the capabilities of the system: *traceability*, *monitoring traceability data*, *handling food safety incidents* and to *buy, share, and sell traceability data*.

[9] https://github.com/solid/solid-spec

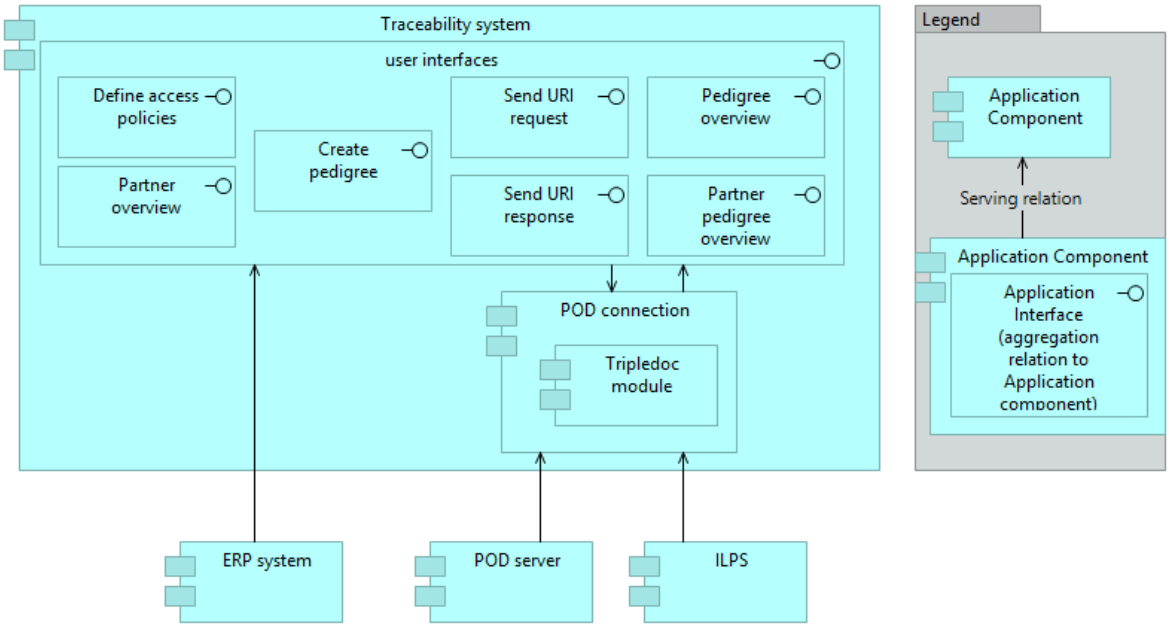


Figure 9: Software engineering diagram of the traceability system. Indicates the three main components: user interfaces and the and POD connection. Three supporting applications are shown.

- The *partner overview* is used to define trade partners. Since Solid is decentralised, the applications must know where to send requests to and must know who has sent a request. Actors realise this in the *partner overview* UI. In this UI, actors enter the WebID and other data describing partners. The application can use this information in later stages to send messages to partners and show which partner has sent a message. Moreover, the application uses the WebID standard during authorisation of data requests[10]. Similar to concurrent web applications where users have a username, WebID creates a unique username, represented as a URI, located at the identity or POD provider. The WebID file contains items like the name and profile picture of a person or organisation.

- Four UIs create the data required for traceability. Actors use the *create pedigree* UI to create a new pedigree and fill it with traceability data. Actors can add additional data to a pedigree in the *pedigree overview* screen. The *pedigree overview* can import data from existing systems into the traceability system, multiple solutions are possible and depend on the presence of an ERP system. Once the data are loaded onto the application, it can be saved to a POD like all other data.

[10] https://www.w3.org/2005/Incubator/webid/spec/identity/

– **With ERP system present**

1. Query the ERP systems' database.
2. Use the API of the ERP system.
3. Transfer the ERP systems' database to the POD provider. From where it can be accessed by the application.

– **Without ERP system**

1. Enter data into the traceability system by hand. This is comparable to filling in a paper form.
2. Using Barcode scanners or IoT devices that register the traceability data required.

Two additional UIs are present to share pedigree URIs, the *send URI request* and *send URI response*. The application uses the pedigree URIs as pointers to retrieve traceability data of partners. The actors use the *send URI request* UI to request a pedigree URI from a partner. The actors use the *send URI response* UI to respond to the requests received from trade partners. The (manual) *send URI request* and *send URI response* UIs become obsolete when pedigrees are sent and received automatically when scanning consignments. However, these UIs will remain in use by the actors that lack the hardware for automation.

- Actors use the *pedigree overview* and *partner pedigree overview* to visualise the traceability data. Actors use the *pedigree overview* to look at the data in their pedigree. They use the *partner pedigree overview* to view additional data in pedigrees of upstream and downstream products. The pedigree URIs provide the access points for this. These overview UIs realise the capabilities to *handle food safety incidents* and *monitor traceability data* shown in figure 8.

- Actors use the *define access policies* UI to sell and share data. In this UI, the actors can define the access rights for each partner. Actors define the contract details like data-resharing, copying, privacy and payment trough other means. This UI only defines the right to access for partners. Actors use the *partner pedigree overview* to view the data once they negotiated access.

  For authorisation, the Web access control(WAC) is used[11]. WAC generates an authorisation document for each container. The authorisation document contains actors (identified by their WebID) that are assigned rights and files to access in that folder. The WebID-TLS protocol[12] authenticates their WebIDs and thus access rights of users with their identity provider. Moreover, the WebID-TLS protocol ensures a secure connection between the applications and PODs.

[11] https://github.com/solid/web-access-control-spec
[12] https://www.w3.org/2005/Incubator/webid/spec/tls/

To summarise this paragraph the following components have been identified to realise the transition from the baseline architecture to the target architecture: seven UIs, a POD connection component, a POD server and the ILPS.

5.2.2 Data architecture

The goal of the data architecture is to define the data that enables the business architecture. Additionally, it defines which standards are used and how data are transferred from the current situation (baseline) to the to be situation (target architecture). In figure 11, the data used by the traceability system can be seen.

To realise the target business architecture, three categories of data are required for the traceability system at an actor: the traceability data, data of partners and access control data.

- **Traceability data**

  The traceability data is required to realise the business goals of *handling food safety incidents* and *monitoring traceability data*. The traceability data is used by the *pedigree overview*, *partner pedigree overview*, *send URI request* and *send URI response* interfaces. The traceability data is captured in the form of a pedigree. A pedigree is the digital representation of physical products at a supply chain actor. A pedigree can describe sizes ranging from a box, to a pallet, to a truckload. Each pedigree should be filled with at least the following items:

  - Link(s) to previous pedigrees for upstream traceability. This means that a record is present that can identify the pedigrees of ingredients used in this pedigree.
  - Link(s) to pedigrees for downstream traceability. This means that a record is present that identifies in what produce this product is used as an ingredient.
  - An item that describes the physical product a pedigree describes.
  - A set of EPCIS events that describe the actions an actor has done to the product. This should capture the internal traceability of an actor. Examples of this for a miller include: mill wheat, put flour in a flour bag, aggregation of flour bags on a pallet.
  - An identifier that relates the physical product to the pedigree. This can be bar-codes, QR-codes and stamps on products or containers.

  Out of these records the following concepts and relations can be defined:

  - **Pedigree** The pedigree concept is the digital representation of a consignment. The *pedigree* captures the traceability data of one consignment and one actor.
  - **Has send pedigree** A relation between two *pedigrees*. Actors use this relation when they sent an pedigree and corresponding consignment to a downstream actor.

- **Has received pedigree** A relation between two *pedigrees*. Actors use this relation when they received an pedigree and corresponding consignment from an upstream actor.
- **Product or service** The product or service offered by the actor in this *pedigree*.
- **Has product info** A relation between a *pedigree* and *product or service*.
- **EPCIS event** Represents a EPCIS event (Solanki & Brewster, 2014).
- **Has consignment info** A relation between a *pedigree* and *EPCIS event*s
- **EPC** The EPC is a concept that captures the digital identifier of a consignment. The EPC can capture barcodes, QR codes or any other identifier.
- **Has identifier** A relation between a *pedigree* and a *EPC*. Used to link the digital pedigree to the physical world.

- **Partner data**
  The partner data describe the partners of the actor. It is required to determine to which data each partner has access. Moreover, it is required to exchange data and messages between the actors. The *partner overview*, *define access policies*, *send URI request*, *Send URI response* interfaces use this record. For each partner the following data should be captured:

  - A WebID. This is the identification of the partner. Each time a partner makes a URI request or request for data, the system uses the WebID in the process of authorising the requests. Moreover, the WebID can be used to find additional data on the partner like addresses.
  - A name of the partner. This is used to display a name in interfaces instead of the WebID.
  - A list of pedigrees, that the actor has access to.
  - A relation to an *access control group* (see below).

  From these partner records, the following concepts and relations can be defined:

  - **Partner** The concept that represents trade partners of a supply chain actor.
  - **Name** Name is a concept that captures the name of the partner.
  - **Legal name** This is a relation between the *partner* and *name* concepts.
  - **Personal profile document** This concept captures the Web ID of actors.

- **is primairy topic of** This is a relation between the *partner* and the *personal profile document* concepts.
- **member** Indicates the relation between a *partner* and *access control group*.
- **access to** This is a relation between a *partner* and a *pedigree*.

- **Access control group**
  The access control group data objects describe to which pieces of a pedigree an actor has access to. This is used to realise the fine-grained access control. The access control group object should capture contain:
  - A name of the access group. This is used for the convenience of end users.
  - A list of partners that are assigned the access rights of this group.
  - A model of the pedigree data, that captures which part of the pedigree can be accessed by this access group. This should mimic the data model of the pedigree.

  For these access control records, the following concepts can be defined:
  - **Agent group** This is the concept that represents an access control group. It captures actors with similar access rights to *pedigrees*.
  - **Name** Name is a concept that captures the name of the agent group.
  - **Has name** This is a relation between the *name* and *Agent group*
  - **Read** This is a relation between the *Agent group* and pedigree data that indicates this access control group has access to the pedigree data. E.g. the relation *read*, between *agent group* and *EPC* indicates access to the *EPC* of a pedigree.
  - **Has member** This is a relation between the *agent group* and *partner*s.

  Next to the data at the traceability applications, the ILPS needs data for a backup as well. The ILPS should only capture links to the pedigrees, not the data in a pedigree. Hereby, the ILPS can recreate the traceability chain, while allowing the actors to remain in control of their data. For each pedigree, the ILPS should capture:
  - The link to the considered pedigree. This should be used to identify the pedigree of a supply chain actor.
  - The owner of the pedigree. This is used to identify the owner of a (partial) product. So that it is known who has made the product, even when they disappear.
  - The link to downstream pedigrees. This is used to capture the products in which this pedigree was used, and thus backup the downstream traceability

- The link to upstream pedigrees. This is used to capture the ingredients that were used in this pedigree, and thus backup the upstream traceability

These records of the ILPS are translated into the following concepts and relations:

- **ILPS record** A concept that encapsulates one backup record.
- **Backup pedigree** A relation between a *ILPS record* and a *pedigree*
- **Has owner** A relation between a *ILPS record* and a *personal profile document*
- **Previous ILPS record** A relation between two *ILPS record*s. Creates a link to the backup of upstream pedigrees.
- **Next ILPS record** A relation between two *ILPS record*s. Creates a link to the backup of downstream pedigrees

5.3 Technology architecture

Figure 10 shows a detailed model of the functionalities and interfaces that are present. It is related to the software engineering diagram of figure 9, but with additional detail on the functions behind the user interfaces. Because Solid creates-client side apps, functionality happens on a local computer. This means that all *technology events* and *technology processes* (E.g. saving data in a pedigree) are hidden in the functionalities of the UIs. The tripledoc module realises the *read* and *write service* to the PODs. This can be PODs of the organisation itself, the ILPS or partner organisations. Each UI reads data from the pods to present the data to the users. When fields are altered and saved the corresponding processes sends the data to the POD with the use of the Tripledoc module. The POD server provides functions to save data, authenticate and authorise users. The *create pedigree screen* is used to create pedigrees which are later used to create traceability. pedigrees can be viewed and edited in the *edit pedigree screen*. If it is altered, the pedigree is saved. In the *view pedigree chain* screen, all available data of the pedigree and precursors will be shown. This screen can be used when a food safety incident happens. In the *partner screen*, actors can assign a partner to an access control group and grant a partner access to a pedigree. This triggers the *save partner* and *update ACL files* functions. The *request response screen* and *pedigree URI request screen* are used in the creation of Linked pedigrees. The *pedigree URI request screen* provides a place to manually create a request at the partner. The *request response screen* provides a list of all requests for pedigree connections made by actors. On this screen, an actor can couple their local pedigree to the pedigree of the downstream actor. In the *data request screen*, a request can be made for additional access rights to a pedigree. The *access control groups* provide a location where new access control groups are made, and the access rights for groups defined. Finally, the *import data screen* is used to manually import data from production processes into the system.

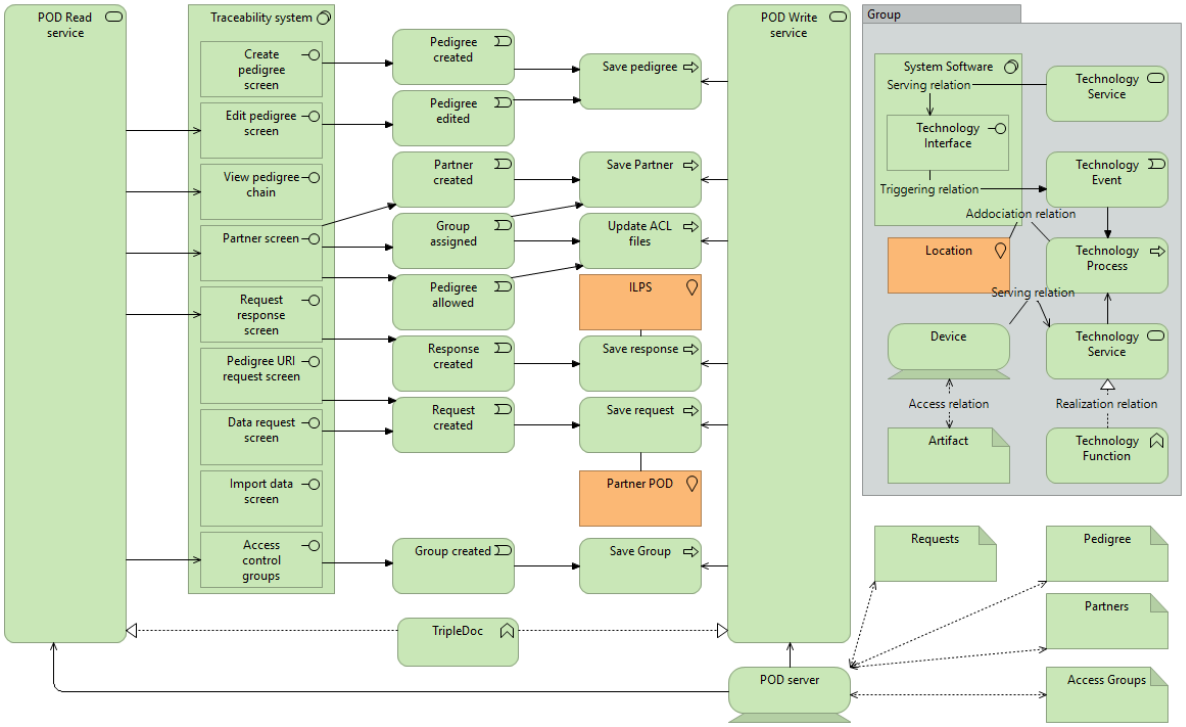


Figure 10: technology architecture of the traceability system. Indicates the interfaces and functionalities provided.

5.3.1 ViewPoint: storing data

The data model described in section 5.2.2 is stored on the Solid pods in RDF. Each object identified in the data architecture gets their URI. The result of this can be seen in figure 11.

The URIs of the pedigree are based on the OntoPedigree design pattern. Actors can extend the data model of the pedigree to their wishes according to the OntoPedigree design pattern. The subject *pedigree* is identified with a unique URI called the pedigree URI. The links to previous pedigree URIs are saved using the *ped:hasReceivedPedigree*[13] predicate and a pedigree URI of the upstream pedigree. The links to downstream pedigrees are saved using the *ped2:hasSendPedigree* predicate and a pedigree URI. The *ped2:hasSendPedigree* is a new predicate created as an opposite of *ped:hasReceivedPedigree*. The *ped:hasProductInfo* predicate and *gr:productOrService*[14] are used to describe the product in a pedigree. The EPCIS events, or actions of an actor on the product are saved using the *ped:hasConsignmentInfo* predicate and *eem:SetOfEPCISEvents*[15]

[13]Prefix ped = http://purl.org/pedigree
[14]Prefix gr = http://purl.org/goodrelations/v1
[15]Prefix eem = http://purl.org/eem

object. Finally, the relation with a physical product is saved using the *ped2:hasIdentifier* and the *ped2:EPCCode* object, which stores an EPC, QR-code or any other identifier.

The URIs of the partner data are based on various available predicates of ontologies. The name of the partner is saved as a String, with the *gr:legalName* predicate. The WebID of the partner is saved using the *foaf:isPrimaryTopicOf*[16] predicate. Then a list of pedigrees this actor has access to is saved using the *acl:accessTo*[17] predicate. Finally, the pointer to the access group this partner belongs to is saved using the *foaf:member* predicate. The URIs of the access control group should largely mimic the ones of the pedigree. However, in stead of using the appropriate predicate, the *acl:read* predicate is used. Once an *acl:read* and object of the pedigree is present, this actor group has access to this triple. Furthermore, the name of the access group is saved using the *ped2:hasName* predicate.

[16]Prefix foaf = http://xmlns.com/foaf/0.1/
[17]prefix acl = http://www.w3.org/ns/auth/acl

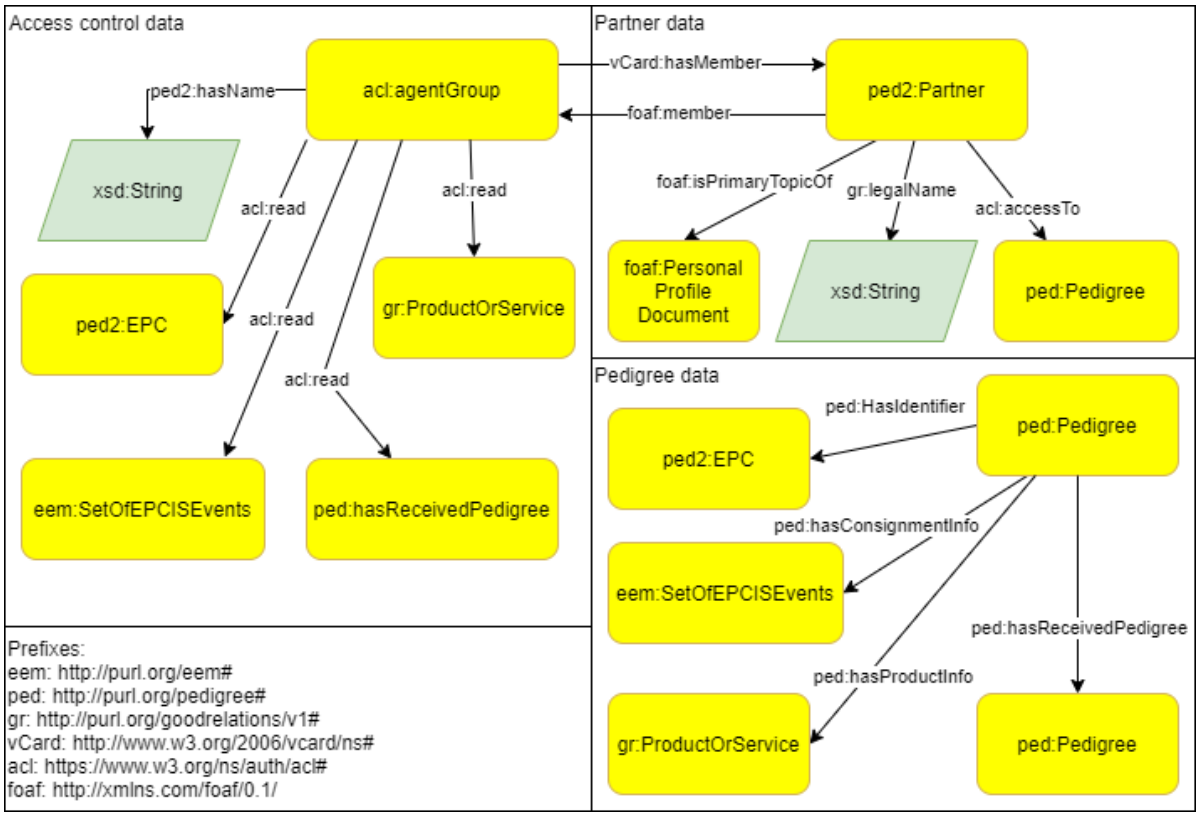


Figure 11: Structure of the data at the supply chain actors. Split into three blocks: the pedigree data to capture traceability data, the data describing a partner, and the data used for access control. The ped2 prefix indicates new objects and predicates. Concepts and relations correspond to the ones defines in the data architecture

In the current state of Solid, the maximum level of granularity is a file. In solid, data are stored in files which are stored in containers. Solid containers can be compared to folders(with files in them) on a windows machine. Each container can have an access control file. In this access control file, access to the files for that container are defined. Therefore, the maximum granularity is a file.

To refine the granularity, the pedigree data are spread out over multiple containers and turtle files. The pedigree URI locates the main container of the pedigree. The pedigree URI is the digital identifier of the product. This container has (sub-)containers for each object in OntoPedigree. The sub-containers are named after the subjects they represent and contain. E.g. the pedigree is located in example.com/data/pedigree1/. The hasReceivedPedigree triples are saved in the turtle document located at example.com/data/pedigree1/hasReceivedPedigree/.
The EPCIS event triples are saved in example.com/data/pedigree1/hasConsignmentInfo/EPCISEvent.ttl. The triples are saved in a turtle document and contain only one predicate type.

E.g. the EPCISEvent.ttl only contains triples of the *ped:hasConsignmentInfo* predicate and of object type *eem:EPCISEvent*.

The other data records for the access control policies and partners are saved in a file for access control policies and for partners in "normal" RDF. These do not require the separation over multiple containers because they are not shared with other actors.

To achieve interoperability, the pedigree folder contains an additional turtle document that keeps track of the triples used in the pedigree, their location, and the local version of the OntoPedigree ontology used. This document is referred to as the pedigree tracker. An example tracker document with 2 triples is shown in listing 1. The pedigree tracker file does not contain any traceability data. This can be seen in the example pedigree tracker. Other Linked pedigrees applications use this file to find the location where the data resides. Moreover, it can be used by applications to find the structure of the data, so that interoperability between the different versions of OntoPedigree can be ensured. This can be seen in figure 13, where two possible versions of the OntoPedigree domain model are shown. 21a shows the original version of OntoPedigree. While 21b, shows another version, where the set of EPC is connected directly to the pedigree. When an actor searches for the EPC code of products, they can search for the predicate (eem:AssociatedWithEPCList) and object(eem:setOfEPCs) of their interest, and fetch the corresponding data. Listing 1 corresponds to a local version of the domain model shown in figure 21a.

```
<pedigree URI> <rdf:type> <ped:pedigree>;
              <ped:hasConsignmentInfo> <example.com/data/pedigree1/
                   EPCIS>;
              <eem:associatedWithEPCList> <example.com/data/pedigree1
                   /EPCs>;
              <ped:hasReceivedPedigree>
                   <example.com/data/pedigree1/hasReceivedPedigree>.
<example.com/data/pedigree1/EPCIS> <type> <eem:SetOfEPCISEvents>.
<example.com/data/pedigree1/EPCs> <type> <eem:setOfEPCs>
<example.com/data/pedigree1/hasReceivedPedigree> <type> <pedigree>.
```

Listing 1: Structure of the pedigree tracker document. Shows where the actual data of the pedigree resides.

The data records at the ILPS are relatively simple. The ILPS only captures pedigree URIs, no data that fill the pedigrees. This is done to keep supply chain actors in control of their data while enabling a backup mechanism. The structure of the data records at the ILPS is similar to a double linked list and blockchain block. The ILPS has a *ILPS record* for each pedigree URI. In the same record, it saves: a link to the next(downstream pedigree URI) *ILPS record*, a link the previous(upstream pedigree URI) *ILPS record*, and the owner of the pedigree. This is graphically illustrated in figure 12. As this figure shows, there is no actual data of supply chain actors at the ILPS. This ensures that actors remain in control of their data.

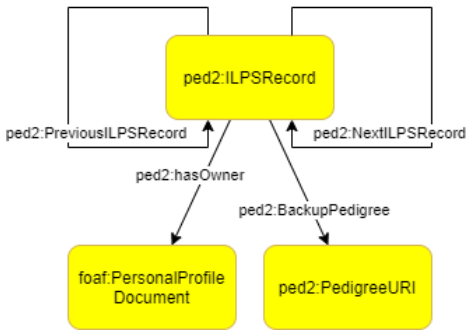


Figure 12: Structure of the data at the ILPS. The ILPS creates a chain of *ILPS records* that contain a pedigree URI and a WebID of the owner. Concepts and relations correspond to the ones defined in the data architecture.

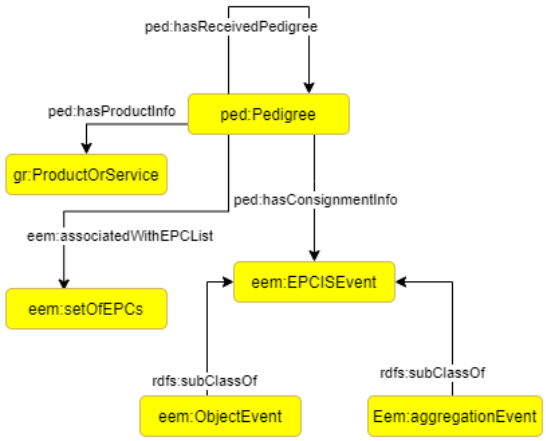
Moreover, all the URIs used are relative. E.g. the ped:hasConsignmentInfo subjects and objects in the example pedigree tracker document of listing 1 should be "./data/pedigree1/EPCIS" instead of "https://example.com/data/pedigree1/EPCIS". In this way, the pedigrees and access control files can be transported to another POD without problems. If the first part of the URI would be included(https://example.com), all the URIs in the pedigrees and access control files would point to the wrong location once data are transferred to another POD. The advantage of relative URIs, is that the URIs can remain the same when switching to another POD provider. Changing from POD provider only requires an update to the pedigree URIs a partner has.

5.3.2 Viewpoint: Creating traceability

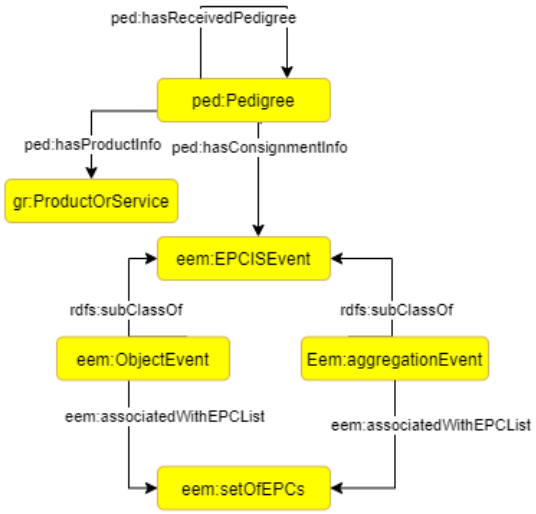
To create traceability between actors, the pedigree URIs must be shared between supply chain actors. The protocol for this is presented in figure 14. It is similar to the protocol of LP, but with additional messages to ensure the upstream and downstream link.

1.1 The protocol starts at A when he sends a consignment to B.

1.2 Upon arrival of the consignment at B, B registers the consignment and makes a request called the *pedigree request*. For this B generates an empty pedigree in its POD, and grants A read rights to the tracker document and append rights to the *ped:hasReceivedPedigree* container of the empty pedigree. The pedigree request of B is a new turtle document saved in the inbox of A. It contains a single subject with four statements as shown in listing 2. The EPC is used to identify the correct product at the upstream partner. The URI of B's pedigree is used in the establishing of the links between pedigrees. The sender is used to know who the sender of the pedigree request is, so that the request can be authenticated.



(a) A possible implementation of the OntoPedigree design pattern. The EPCs of products are places in the *eem:setOFEPCs*, which are placed directly into the pedigree.



(b) A possible implementation of the OntoPedigree design pattern. The EPCs of products are placed behind the EPCIS events.

Figure 13: Two possible implementations of the OntoPedigree design pattern.

```
<any subject name> <https://www.w3.org/TR/rdf-schema/#ch_type>
      <https://linkedpedigrees2.com/PedigreeRequest>;
<https://linkedpedigrees2.com/identifier> <String containing
    the EPC>;
<https://linkedpedigrees.com/pedigree> <URI of B's pedigree>;
<https://linkedpedigrees2.com/sender> <profile document of B>.
```

Listing 2: Structure of a pedigree request.

2.1 A can send a *pedigree response* to B after the *pedigree request*. Once A acknowledges that the pedigree request is valid, he saves the received pedigree URI of B at the corresponding pedigree, using the *ped:hasSendPedigree* predicate. A uses the pedigree URI received from B to send their URI response. This is done by creating a new turtle document in the hasReceivedPedigree folder. This document contains a single triple: *any subject*; *"https://linkedpedigrees.com/hasReceivedPedigree"*; *A's pedigree URI.A* saves this using the PUT method, A got access to that during the pedigree request of B.

2.2 and 2.3 Once an actor receives a pedigree, they will send both URIs to the ILPS. Similar to URI requests, the message creates a new turtle document in the inbox of the ILPS. The ILPS uses both messages as a validation. When only message 2.3 would be present, B is reliant on A to send the correct data to the ILPS. For example, A could send false data to the ILPS, that would accuse B of a food safety incident, while he has done nothing wrong.

Or A could not send the message at all, while B trusts A to send it. This message contains: the pedigree URI of B, the pedigree URI of A, and the WebID of B. This message is structured as the statements in listing 3:

```
<any subject name> <https://www.w3.org/TR/rdf-schema/#ch_type>
     <https://linkedpedigrees.com/ILPSmessage>;
<https://linkedpedigrees.com/actorPedigree> <own pedigree URI
    >;
<https://linkedpedigrees.com/usedIn> <pedigree URI of
    downstream partner>;
OR
<https://linkedpedigrees.com/derivedFrom> <pedigree URI of
    received goods>;
<https://linkedpedigrees.com/sender> <profile document the
    sender>.
```

Listing 3: Structure of a message to the ILPS.

3.1 For the more complex production processes, multiple messages are sent to the ILPS. For example, if A has multiple pedigrees for their ready-made lasagne, the sauce used and the pasta sheets used. (The ready-made lasagne - pasta sheets) and (ready-made lasagne - lasagne sauce) pedigree links have to be sent to the ILPS as well. This ensures that the internal traceability is intact. This message contains the pedigree URIs of the actor and his WebID.

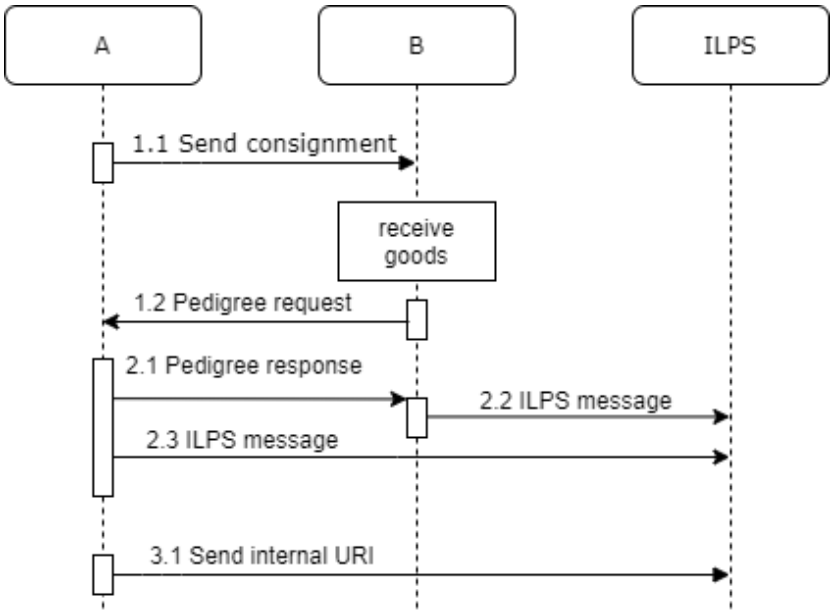


Figure 14: Adjusted communication protocol of LP. Indicates how pedigree URIs are exchanged and the product links between pedigrees are established.

Exchange of additional data can be done directly. Thanks to tripledoc and the acquired pedigree URIs, actors can directly view data from partners they have access to. However, it may be possible that an actor wants to see more than his access rights currently allow. For this, the actors should contact each other.

5.3.3 Viewpoint: access control

The previous viewpoint introduced the communication models for URI exchange and data exchange. This section will explain the access control in detail.

In Solid, each container has a document which determines the access rights to that container. If a container does not have such a document, the parent containers' document is used. This process repeats until an access control document is found. The root folder must contain an access control document. The access control document defines: 1) actors, 2) their rights and 3) the affected resources or containers. Trading partners have the same access rights for each consignment. The access control is based on the ontology(designed with the OntoPedigree design pattern), following the logic of ontology based access control (Brewster et al., 2019). OBAC is chosen for access control because: 1) it extends on a role based access control framework, which is present in Solid, 2) it creates a more efficient way of managing access to similar resources, and 3) creates a better overview. For OBAC to be implemented it must satisfy four requirements. These requirements as well as their implementation are shown in table 15. Although the data are not accessed by referring to the structure of the (meta) data graph, the benefits of OBAC for end-users are still realised.

At this point, the actors should define who has access to which part of their data model. For this, the access control UI presents an interactive graph structure of the data model (based on OntoPedigree). Users click on the nodes in the graph structure. By clicking on the nodes, access is granted or denied to that specific node. This is extended by literal values that are forbidden as well. Now, the access policies must be translated to the ACL files of the corresponding pedigrees. The example access control file in listing 4 indicates the access rights of *farmer1* to the hasReceivedPedigree triples in the pedigree of the *trader*. It shows that access rights are granted to one actor, per node in the data model. The trader (owner of the data) has all the access rights (read, write, and control).

```
@prefix n0: <http://www.w3.org/ns/auth/acl#>.
@prefix lin: <https://linkedpedigrees/>.

:trader
    a n0:Authorization;
    n0:accessTo lin:hasReceivedPedigree, <./>;
    n0:agent <https://trader.example.com/profile/card#me>;
    n0:default <./>;
    n0:mode n0:Control, n0:Read, n0:Write.
:farmer1
    a n0:Authorization;
```

```
n0:accessTo <./>;
n0:agent https://farmer1.solid.community/profile/card#me;
n0:default <./>;
n0:mode n0:Read.
```

Listing 4: Example ACL file granting farmer1 access to the hasReceivedPedigree triples folders.

It is chosen to assign the right to view a pedigree per actor. Even tough Solid can use acl:agentClass[18] to indicate access per group rather than per actor. Using acl:agentClass creates a vulnerability where every actor in that group has access to a pedigree, while one a specific one is meant to have access. This creates the opportunity to control who has access to a specific part of the pedigree.

[18]https://www.w3.org/ns/auth/acl#agentClass

OBAC requirement	OBAC implementation
"Metadata needs to be assigned to raw object data prior to access." (Brewster et al., 2019)	By the implementation of the data in agri, the data are assigned metadata before access. Moreover, the pedigree tracker document provides a location where metadata of the pedigrees can be found.
"The metadata scheme adheres to an ontology: it is hierarchically structured, with meaningful (interpretable, semantic) relations between nodes (concepts) and reflects domain knowledge." (Brewster et al., 2019)	By using the OntoPedigree design pattern, every actor can design their own domain ontology for traceability. By using OntoPedigree and agri, it is hierarchically structured and has meaningful relations between the concepts.
"Access to the object data occurs through the metadata, with the possibility of defining access for a given person or role to specific layers (strata) in the metadata." (Brewster et al., 2019)	Access to objects happens directly at the data, not through the meta data. Defining access for a person or role based on the structure of the data happens in the access control UI. This UI an image that represents the ontology used by the actor. By clicking on one of the concepts, access to this concept can be changed. Additionally, specific values can be given that are forbidden for an actor to see. The ACL files are updated according to the data entered in the access control UI and define the access to triples of one predicate or value.
"Access to object data and re-usability of (meta)data is determined by referring to the structure of the metadata graph, the contents of the nodes, or both." (Brewster et al., 2019)	Access to the data are determined by the ACL file that accompanies a file that contains triples. The values(access or not) in the ACL file are determined in the access control UI, where access to a node can be defined based on the node or the values in a node.

Table 15: This table shows how the requirements of OBAC are implemented in the artefact.

6 Treatment validation

This section contains the third element in the design cycle. The goal of the treatment validation is "to predict how an artefact will interact with its context, without actually observing an implemented artefact in a real-world context" (Wieringa, 2014). The results show if the artefact contributes to the stakeholder goals. Central to the validation are four categories of validation questions: effect, trade-off, sensitivity and requirement satisfaction questions (Wieringa, 2014). In the effect category, the artefact is exposed to a model of the context and answers questions related to the effects produced by exposing the artefact to a model of the context. In the trade-off category, alternative artefacts are exposed to the same context and compared. The sensitivity category introduces the artefact to a different context. In the requirements satisfaction category, the artefact is reviewed for the satisfaction of the requirements. The final section validates if the artefact contributes to stakeholder goals.

6.1 Validation method

To test the designed treatment a proof of concept(POC) of the architecture was made. The POC was used as a model of the artefact for the validation of the created architecture. The POC is introduced in section 6.2, a detailed version can be found in appendix A. After this, the four categories of validation questions structure this section. The effect questions are used to validate the artefact for its ability to contribute to handling food safety incidents. To do this, scenarios of (fictional) food safety incidents have been used as a model of the context. The scenarios are used to validate that the artefact can 1) create traceability, 2) handle disappearing actors, 3) prevent access to private data, and 4) import data to the system. This is based on the functional requirements. Because the context is the same for effect questions and trade-off questions, these have been grouped in the same section. To compare the results of the artefact with other artefacts, the commercially available systems were used. The requirement satisfaction section is concerned with the requirement satisfaction questions. The introduction of the artefact to a wider context is the focus of the sensitivity questions section. To do this, values of the POC have been taken and increased to a value where the entire dutch supply chain uses the system.

6.2 Proof of concept

To show that the presented architecture in the previous section can be built, a POC has been made. The proof of concept is used in the remainder of this chapter to validate the architecture. The POC was implemented with the use of a nodeJS server[19]. The Solid community server has been used as a POD provider. And built by extending the example solid application (*Writing a Solid application*, n.d.). The POC has a UI that for every UI listed in the technology architecture (figure 10). The exceptions being the *data import screen*, which has

[19] https://nodejs.org/en/

been made part of the *edit pedigree screen*, and the *data request screen*, which has not been implemented. Because this can be done through e-mail or a phone call. The UIs can be accessed by the bar with buttons on the top of figure 15. Moreover, this figure gives an overview of the pedigrees currently present at an actor. Figure 16, presents the UI that shows the traceability records of a batch ready-made lasagne that is used in the validation. Another important UI to present here is the access control UI of figure 29. This interface shows a graphical representation of the OntoPedigree version used in the POC. This figure is interactive and creates the opportunity to define the access to triples of a specific kind. Further details on the POC and an explanation per UI can be found in appendix A.

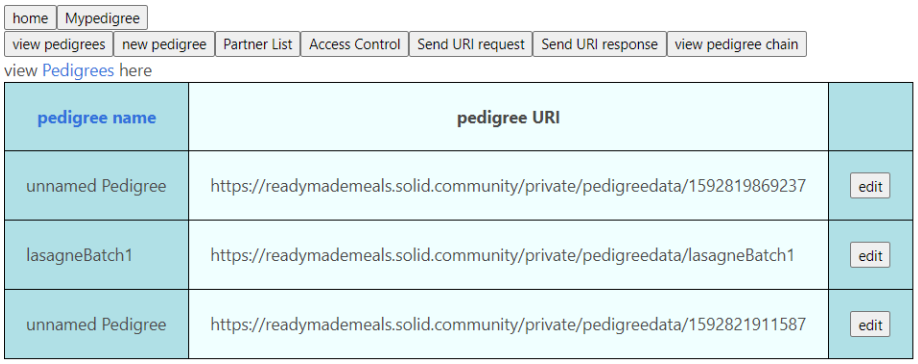


Figure 15: View pedigree interface of the proof of concept. grants an overview of the pedigrees in the owners' POD

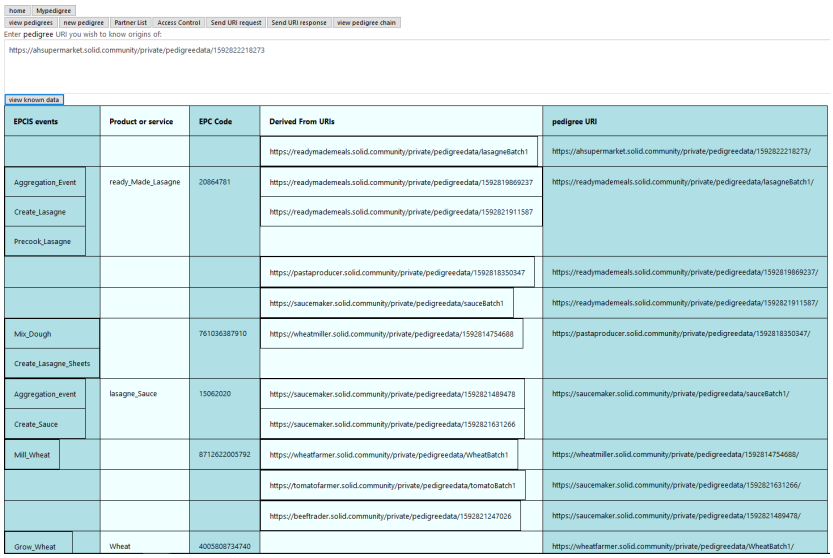
6.3 Effect & trade-off questions

To create a context for the effect and trade-off category, multiple scenarios have been made. The scenarios are based on the horsemeat scandal with some adjustments (E.g. an actor disappearing). As the ready-made lasagne is used as a motivating example throughout this thesis, this is also used as a model of the context for the validation. To create the scenarios, a Solid profile has been made for every actor in figure 3a. Then the actors create a single batch of ready-made lasagne in the POC. To model the steps taken by every actor, each actor has EPCIS events in its pedigree simulating the business processes in figure 3b used to create a batch of ready-made lasagne. In this scenario, the slaughterhouse is named Draap.

6.3.1 Traceability scenario

The goal of this scenario is to validate that the artefact is capable of creating traceability and thereby finding clues for the origin of a food safety incident. Traceability is considered validated when the entire lifecycle of a product can be found. The data for the scenario has been created in a way that would happen

in the real context as well. First, the wheat farmer has created a pedigree. Then it has added the EPCIS events *Grow_Wheat and Harvest_Wheat* described as business processes in figure 3b. Then the wheat is sent to the wheat miller. The wheat miller sends a request for the pedigree URI to the wheat farmer. In turn, the wheat farmer sends a response. The wheat miller sends the flour to the pasta producer. Each actor assigns access to the pedigree to their wishes and includes the supermarket to have access to at least the hasReceivedPedigree triples. This process repeats until the supermarket is reached. The supermarket receives a message from food safety authorities that the ready-made lasagne is contaminated with horsemeat. At this point, the supermarket enters the pedigree URI of the ready-made lasagne and finds the results of figure 16. Figure 16b contains a pedigree of the beef trader, containing a hasReveivedPedigree link to Draap. With the use of this traceability overview, the supermarket has an indication who could have been the contaminant. This scenario validates that the artefact is capable of creating traceability. Additionally, it validates the linked traceability requirements, as each pedigree is located in a different POD, and has links to other parts of the data available.



(a)

(b)

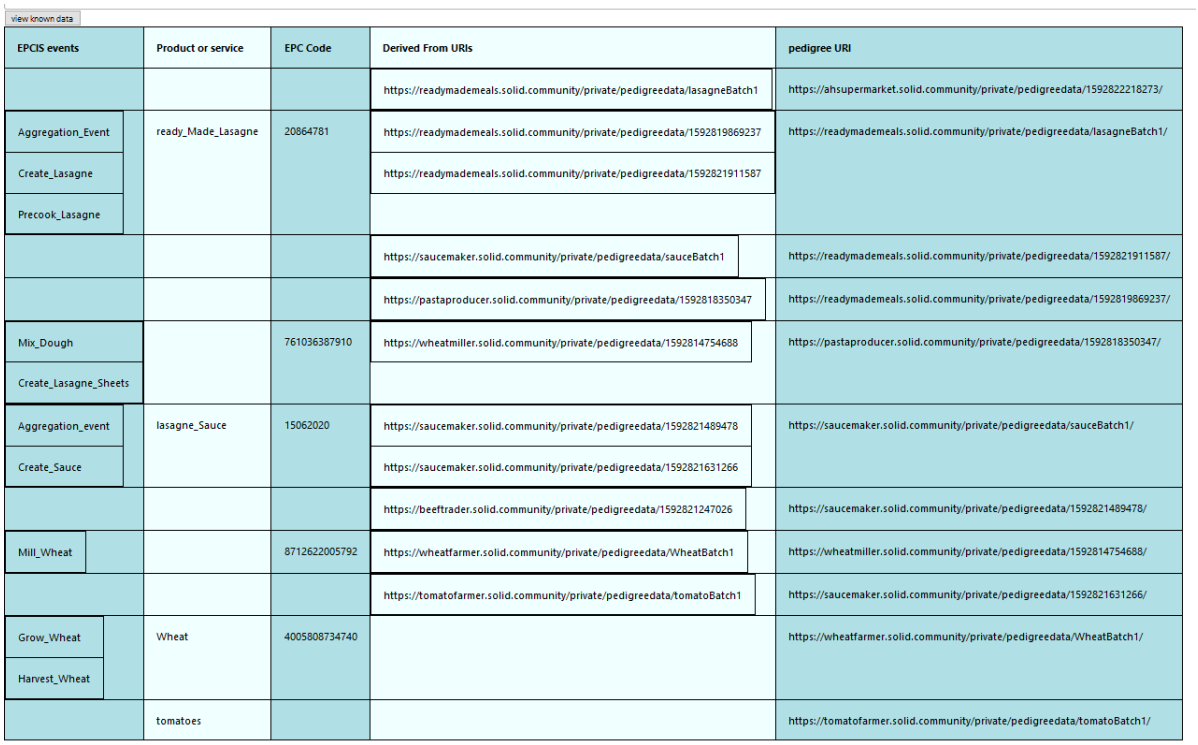
Figure 16: Pedigree chain view able by the supermarket. Split in two because it was too big for one screen. (a): First half of the results view-able for the supermarket. (b): Second half of the results view-able for the supermarket.

In comparison to the other traceability initiatives (section 4.1), the artefact has a slower response time than IBM food trust (2.2 seconds). Although the other systems do not provide the timing of traceability, they are expected to be faster than the artefact. This is because they rely on a centralised location of the data to query while the artefact has to query every POD in the traceability chain. Similar to other systems all can create traceability. In comparison to paper-based traceability, the artefact is faster.

6.3.2 Backup traceability scenario

The goal of this scenario is to validate that the system can recreate the traceability chain, even when an actor disappears. Backup traceability is considered satisfied when actors behind the disappeared actor can be reached again. This scenario builds on the traceability scenario by removing the access rights of the supermarket from the beef importers' pedigree. The results of the traceability can be seen in figure 17. The results are similar to the previous ones, except for the missing pedigree information of the beef trader and further upstream. To find the missing traceability data, the last known pedigree is searched for at the ILPS. In this case the, last known pedigree URI is: "https://beeftrader.solid.community/private/pedigreedata/159282
This URI can be searched for in the ILPS (figure 33, row two). The ILPS tells the next pedigree to be: "https://draap.solid.community/private/pedigreedata/1592820713661".
If the supermarket plugs this pedigree into their pedigree chain interface it generates the results in figure 18. This shows the missing upstream information from the cattle farmer and the slaughterhouse. This scenario validates the backup traceability.

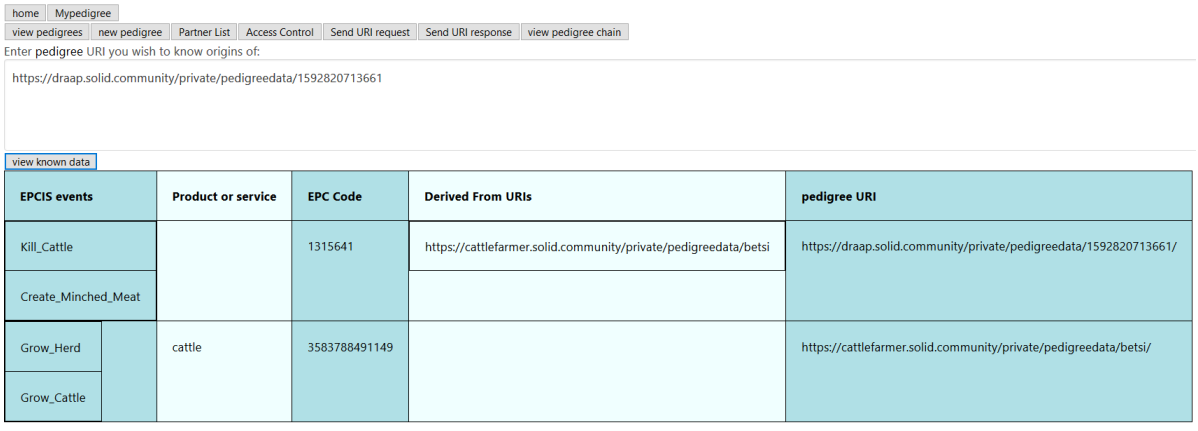
The other artefacts in the same context would not respond to this problem. As three of the other systems rely on the blockchain, removing a data-record is simply not possible. If one record is deleted, the entire blockchain is no longer reliable. For Traces, there is most likely a backup on another location in case data are unavailable. Therefore all other artefacts are capable of realising backup traceability.



view known data

EPCIS events	Product or service	EPC Code	Derived From URIs	pedigree URI
			https://readymademeals.solid.community/private/pedigreedata/lasagneBatch1	https://ahsupermarket.solid.community/private/pedigreedata/1592822218273/
Aggregation_Event	ready_Made_Lasagne	20864781	https://readymademeals.solid.community/private/pedigreedata/1592819869237	https://readymademeals.solid.community/private/pedigreedata/lasagneBatch1/
Create_Lasagne			https://readymademeals.solid.community/private/pedigreedata/1592821911587	
Precook_Lasagne				
			https://saucemaker.solid.community/private/pedigreedata/sauceBatch1	https://readymademeals.solid.community/private/pedigreedata/1592821911587/
			https://pastaproducer.solid.community/private/pedigreedata/1592818350347	https://readymademeals.solid.community/private/pedigreedata/1592819869237/
Mix_Dough		761036387910	https://wheatmiller.solid.community/private/pedigreedata/1592814754688	https://pastaproducer.solid.community/private/pedigreedata/1592818350347/
Create_Lasagne_Sheets				
Aggregation_event	lasagne_Sauce	15062020	https://saucemaker.solid.community/private/pedigreedata/1592821489478	https://saucemaker.solid.community/private/pedigreedata/sauceBatch1/
Create_Sauce			https://saucemaker.solid.community/private/pedigreedata/1592821631266	
			https://beeftrader.solid.community/private/pedigreedata/1592821247026	https://saucemaker.solid.community/private/pedigreedata/1592821489478/
Mill_Wheat		8712622005792	https://wheatfarmer.solid.community/private/pedigreedata/WheatBatch1	https://wheatmiller.solid.community/private/pedigreedata/1592814754688/
			https://tomatofarmer.solid.community/private/pedigreedata/tomatoBatch1	https://saucemaker.solid.community/private/pedigreedata/1592821631266/
Grow_Wheat	Wheat	4005808734740		https://wheatfarmer.solid.community/private/pedigreedata/WheatBatch1/
Harvest_Wheat				
	tomatoes			https://tomatofarmer.solid.community/private/pedigreedata/tomatoBatch1/

Figure 17: Results of the pedigree chain view-able by the supermarket when the beef trader revokes access or disappears.



home | Mypedigree

view pedigrees | new pedigree | Partner List | Access Control | Send URI request | Send URI response | view pedigree chain

Enter pedigree URI you wish to know origins of:

https://draap.solid.community/private/pedigreedata/1592820713661

view known data

EPCIS events	Product or service	EPC Code	Derived From URIs	pedigree URI
Kill_Cattle		1315641	https://cattlefarmer.solid.community/private/pedigreedata/betsi	https://draap.solid.community/private/pedigreedata/1592820713661/
Create_Minched_Meat				
Grow_Herd	cattle	3583788491149		https://cattlefarmer.solid.community/private/pedigreedata/betsi/
Grow_Cattle				

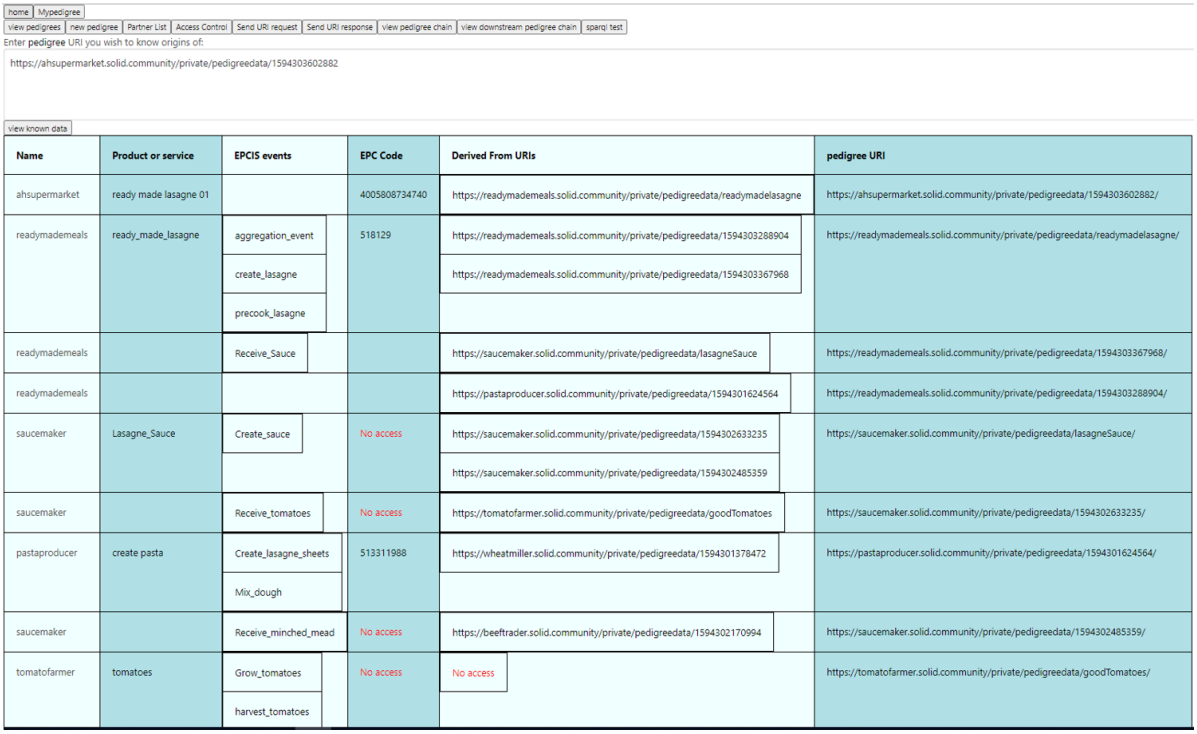
Figure 18: Recovered data retrieved with the help of the ILPS.

6.3.3 Access control scenario

The goal of this scenario is to validate that the system does not give access to data that is considered private by an actor and is considered validated when it does so. In the traceability scenario, the supermarket had access to all the data. In this scenario, the sauce maker and the tomato farmer restrict access to the EPC code of the pedigree. The results are shown in figure 19. It can be seen that the supermarket still has access to some of the data in pedigrees of the sauce maker and the tomato farmer actors. But not to the EPC category. This validates that the artefact is capable of creating fine-grained access control.

In section 2.3, six possible risks of sharing data between actors were identified. All these risks can be mitigated by an actor if the granted access rights are correct. As seen in the access control scenario above, the supermarket could not view any EPC information of the sauce maker. If the sauce maker restricted all access, the ready-made lasagne producer could not gain any new information that realises the risks. These risks could become reality if the sauce maker actor grants full access to their pedigree(s). There are two other possible ways how these risks can occur: the first possibility for these risks to occur is when a data breach happens at the ILPS, POD provider or application provider. The second possibility for these risks to occur is when a trusted actor with the correct access rights publishes the data they have access to. This requirement is considered validated.

The other traceability systems do not appear to be capable of defining access on a fine-grained scale. Only IBM food trust could be able to do this. However, this cannot be proven or falsified at this without access to the application.



Name	Product or service	EPCIS events	EPC Code	Derived From URIs	pedigree URI
ahsupermarket	ready made lasagne 01		4005808734740	https://readymademeals.solid.community/private/pedigreedata/readymadelasagne	https://ahsupermarket.solid.community/private/pedigreedata/1594303602882/
readymademeals	ready_made_lasagne	aggregation_event create_lasagne precook_lasagne	518129	https://readymademeals.solid.community/private/pedigreedata/1594303288904 https://readymademeals.solid.community/private/pedigreedata/1594303367968	https://readymademeals.solid.community/private/pedigreedata/readymadelasagne/
readymademeals		Receive_Sauce		https://saucemaker.solid.community/private/pedigreedata/lasagneSauce	https://readymademeals.solid.community/private/pedigreedata/1594303367968/
readymademeals				https://pastaproducer.solid.community/private/pedigreedata/1594301624564	https://readymademeals.solid.community/private/pedigreedata/1594303288904/
saucemaker	Lasagne_Sauce	Create_sauce	No access	https://saucemaker.solid.community/private/pedigreedata/1594302633235 https://saucemaker.solid.community/private/pedigreedata/1594302485359	https://saucemaker.solid.community/private/pedigreedata/lasagneSauce/
saucemaker		Receive_tomatoes	No access	https://tomatofarmer.solid.community/private/pedigreedata/goodTomatoes	https://saucemaker.solid.community/private/pedigreedata/1594302633235/
pastaproducer	create pasta	Create_lasagne_sheets Mix_dough	513311988	https://wheatmiller.solid.community/private/pedigreedata/1594301378472	https://pastaproducer.solid.community/private/pedigreedata/1594301624564/
saucemaker		Receive_minched_mead	No access	https://beeftrader.solid.community/private/pedigreedata/1594302170994	https://saucemaker.solid.community/private/pedigreedata/1594302485359/
tomatofarmer	tomatoes	Grow_tomatoes harvest_tomatoes	No access	No access	https://tomatofarmer.solid.community/private/pedigreedata/goodTomatoes/

Figure 19: Results when the supermarket has limited access to the data in the pedigrees of the supply chain actors.

6.3.4 Data import scenario

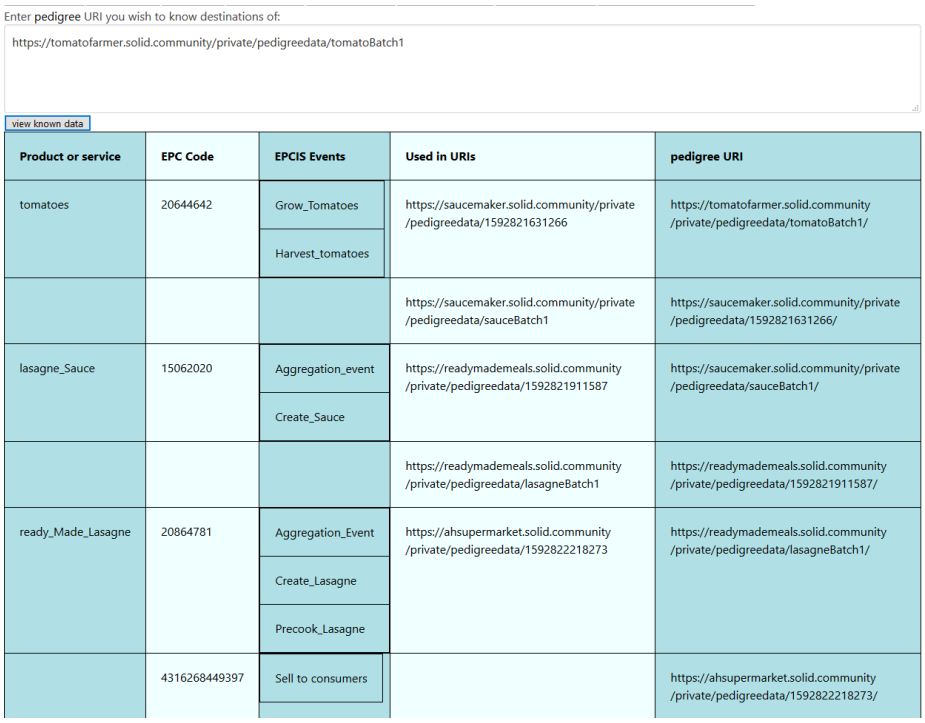
The goal of this scenario is to validate that the artefact is capable of importing data. The ability to import data is considered validated when the system can import data. In the next model of the context, the ready-made meals actor has an ERP system that registers the production of their ready-made lasagne. They are capable of exporting their data in JSON format. When the ready-made lasagne actor has created a new pedigree. The data from the ERP system can be put directly into the pedigree. Listing 5 below shows the ERP data imported used as validation. This Validates that this artefact is capable of reusing data from other systems. As mentioned in section 4.1, all other systems are capable of data reuse.

```
{
    "EPCISEvent":"Create lasagne",
    "derivedFrom":"https://bertolli.com/leftovers_batch1",
    "productOrService":"Ready-made lasagne batch 2",
    "EPCCode":"8713337098627"
}
```

Listing 5: JSON file used to validate the import scenario.

6.3.5 Downstream traceability scenario

The goal of this scenario is to validate that the system is capable of creating traceability. Because traceability includes both upstream and downstream, the downstream traceability has to be validated as well. Downstream traceability is considered satisfied when the most upstream actor can visualise all stages of its product until the supermarket. To validate that the artefact is also capable of creating downstream traceability, the same context as the traceability scenario has been taken. However, in this case, the tomato farmer finds out that his tomatoes are contaminated. To prevent image loss of the supply chain, the tomato farmer enters his pedigree in the *view downstream pedigree chain* and retrieves the data in figure 20. From here, he knows the downstream actors to warn. To let this scenario work, all actors have given access to their pedigrees to the tomato farmer. Next to that, the missing hasSendPedigree predicates have been manually added to the PODs.


Enter pedigree URI you wish to know destinations of:

https://tomatofarmer.solid.community/private/pedigreedata/tomatoBatch1

view known data

Product or service	EPC Code	EPCIS Events	Used in URIs	pedigree URI
tomatoes	20644642	Grow_Tomatoes Harvest_tomatoes	https://saucemaker.solid.community/private/pedigreedata/1592821631266	https://tomatofarmer.solid.community/private/pedigreedata/tomatoBatch1/
			https://saucemaker.solid.community/private/pedigreedata/sauceBatch1	https://saucemaker.solid.community/private/pedigreedata/1592821631266/
lasagne_Sauce	15062020	Aggregation_event Create_Sauce	https://readymademeals.solid.community/private/pedigreedata/1592821911587	https://saucemaker.solid.community/private/pedigreedata/sauceBatch1/
			https://readymademeals.solid.community/private/pedigreedata/lasagneBatch1	https://readymademeals.solid.community/private/pedigreedata/1592821911587/
ready_Made_Lasagne	20864781	Aggregation_Event Create_Lasagne Precook_Lasagne	https://ahsupermarket.solid.community/private/pedigreedata/1592822218273	https://readymademeals.solid.community/private/pedigreedata/lasagneBatch1/
	4316268449397	Sell to consumers		https://ahsupermarket.solid.community/private/pedigreedata/1592822218273/

Figure 20: Results when the tomato farmer looks where his products went to.

6.4 Requirement satisfaction

The goal of this section is to validate that the artefact satisfies the requirements laid down in the requirements, section 3. The requirements are considered satisfied when the fit criterion is satisfied. The scenarios from the previous sections

validate the traceability, the access control requirements, and the value proposition - data reuse. But the traceability - query, value proposition - compelling feature, value proposition - easy to use and trusted software supplier requirements remain and are validated below.

The traceability - query, is partially satisfied. In the traceability scenario, it can be seen that it is possible to query for the available data throughout multiple PODs. Moreover, it is possible to continue searching with the retrieved data from one actor. However, as Solid does not provide a query engine yet, it is not possible to query through the system on demand. A query has to be manually programmed to retrieve the data. Therefore, querying is possible, but not in a "traditional" query language like SQL or SPARQL.

The value proposition - compelling feature requirement can be validated by analogy of the access control requirement. Because the compelling feature is realised by the ability to sell data, the data owner can monetise the access to their data. Instead of a supermarket actor gaining access to the data on request, a buyer is added in the partner list interface(figure 26) and granted access to the pedigrees paid for. This creates a new feature apart from traceability that brings value.

The value proposition - easy to use, can be both validated and falsified depending on the interface used. First, the vision of scanning a barcode at the arrival of a consignment; which automatically registers the product; sends a pedigree request; automatically receives a response URI; creating the traceability chain; satisfies the easy to use requirement. Moreover, end-users only require to use a bar-code scanner. This should cover concerns of organisations to attract new skilled employees to use the system. Employees scan a consignment, compared to filling in paper forms, this should not add to their workload. However, the manual managing of access control for each pedigree may become a tedious job. This does not satisfy the easy to use requirement. On the other hand, the retrieving of pedigree data from upstream actors in a single click does satisfy the requirement and could save time.

The trusted software supplier - reliable system and supplier reputation can be validated by analogy with other applications. For example, take the text editors Microsoft Word, notepad and overleaf. Each of those applications is capable of writing a text. Depending on the demands of the writer a different application can be chosen. Once the chosen application has crashed multiple times, the actor will lose trust in that application and switch to another one. Such a mechanic forces the application providers to build a reliable product or their customers will start disappearing. Because Solid provides the opportunity to easily switch between POD providers and applications. This forces the POD providers and applications providers to build a reliable system. Next to the reliability of the information systems, a similar mechanic is present in the reputation of an application supplier. When a supplier has a negative reputation, actors can easily switch to another application supplier

The trusted software supplier - governance of the system could not be validated. This depends on the willingness of the software companies developing their implementations of the architecture. It may turn out that none of the soft-

ware companies are interested in creating an overarching governance function, and the separate implementations of the architecture create their new standards.

6.5 Sensitivity questions

For the sensitivity of the artefact, of particular interest is the expansion of the context. By an increasing number of users, there are potential bottlenecks. First is the ILPS. During the traceability scenario described, the ILPS has generated 3287 bytes in data. This represents a single batch of ready-made lasagne. Assuming a supermarket receives one such a batch a week and there are 6338 supermarkets in the Netherlands (Rosian & Pustjens, 2019). If the average supermarket has 2000 products (The Albert heijn, ETOS and Gall & Gall combined sell 28500 products on their website (*Feiten en cijfers*, n.d.)) would mean that all supermarkets and their products generate 1.97 TB of data a year at the ILPS. This should not be a problem for a POD provider as they are specialised in storing data. In comparison, Microsoft OneDrive offers 1TB data storage for one user for 4,20 euro a month (*OneDrive-abonnementen vergelijken*, n.d.). This means that storing the data on the ILPS should not be a bottleneck.

Second is the number of requests to the ILPS. Using the same numbers as before uploading 1.97 TB of data over a year should not form a bottleneck either. The number of messages to the ILPS is 18 in total. Using the same numbers as before, an estimated total of 3.2×10^7 messages are sent to the ILPS every year. In comparison, Google handles 3.5×10^{12} searches each day, although this comparison is not entirely fair as this figure is worldwide (*Google Search Statistics*, n.d.). Therefore this figure is adjusted to the percentage of the Dutch inhabitants of the world population(0.223% (CBS, n.d.)), an estimated 7.82×10^9 searches per day remain. Therefore the number of messages to the ILPS should not be a bottleneck either.

A Third bottleneck is the current data structure used to realise the fine-grained access control. To view all the data in a pedigree, 5 requests are sent to a POD. One for the pedigree tracker document, then one for every predicate in the pedigree(currently the productOrService, EPCISEvents, EPCCode and derivedfrom). Once the queries become bigger and span more than one consignment, this could become a bottleneck for a POD. Especially once the data model is extended to contain more than 4 predicates.

Fourth is the amount of data stored in individual actors' PODs. Using the file browser of the Solid community, the pedigree of the sauce maker actor called sauce batch 1(row 6 in figure 17) is said to be 4096 bytes. Assuming that the sauce maker produces 3650 sauce batches a year, the sauce maker would create 0.014 GB of data in their pedigrees a year. Although this number is expected to be higher in reality as the amount of information in a pedigree will be much larger, this should not form a bottleneck.

The fifth bottleneck is the amount of data travelling from the POD to the application and back. Although this has not been measured, it should be noted that the application only loads data when required. Even if the entire 0.014 GB of the third possible bottleneck is passed back and forth for every save, the

application should still function. The application may be slow as the download-/upload speed and the memory cap of machines could be reached. However, the current architecture requires a new data request for every triple in a pedigree. Therefore the amount of data requests to view pedigree could become an issue.

The final potential bottleneck is loading the application at the client. In Solid ecosystem, applications are provided from the server of the application provider and loaded on the local computers of the end-users. This could mean that loading the application to all end-users could create a bottleneck at the start of the production day when all computers are started. For these issues load balancing techniques are available, yet it is unknown if this bottleneck will occur.

One of the assumptions in the context is that users enter correct data. At this moment, there are minimal checks on the input data. This means that an application can be broken with syntactically wrong data. But this should not be too much of an issue once input checks are in place for future applications. A bigger threat for the artefact is the possible false data entry, creating traceability that does not work when it is necessary. E.g. if the sauce-maker actor put in false data about the origin of their products, linking to a beef importer while the true importer sold horsemeat. Then the pedigree chain can no longer be trusted. However, as the URIs are saved at the ILPS, they create proof of forgery. This could work as a deterrent for entering false data.

Once the context grows, it would inevitably become interesting for hackers to attack the system. Fortunately, the decentralised architecture hinders DDOS attacks to the entire system. If for example, the beef trader is under DDOS, all the actors are still able to use their system. The weak points in the architecture are the POD providers if there is not enough diversity in the providers used. A DDOS to a POD provider could cripple a large part of the systems' infrastructure. Another vulnerable location is the ILPS as they are the only centralised location in the architecture. Next to the ability to DDOS, the POD providers, application providers and ILPS are potential interesting locations for hackers to gain sensitive data from the supply chain actors.

As the context grows, the OntoPedigree could become another problem. As the amount of different connected data models grows, there will be a data model used by an actor that does not fit within OntoPedigree. However, as the traceability scenario shows, all data can be entered manually without integration to any system to realise traceability.

6.6 Contribution to stakeholder goals

These scenarios validate that the artefact is capable of contributing to the stakeholder goals of increased food safety. The increased food safety is realised by the ability to have early detection in place. This is validated by a side note of the first scenario. In this scenario, the supermarket could have looked at this data at any moment. Before putting the ready-made lasagne on the shelf, it could have seen beef coming from the Draap company, raising alarms. The generation of the results in figure 16 cost about five seconds. This validates the realisation

of the better response time goal. As organisations can warn the supply chain partners almost directly.

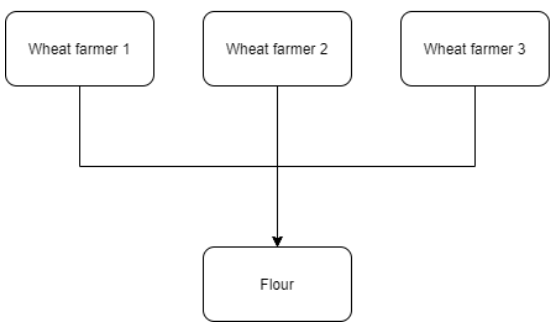
The new revenue streams where traceability data can be sold has been validated in the value proposition - compelling feature. The increased customer satisfaction goal is realised by the ability to share production information with customers as shown in the traceability scenario. The increased market share is realised by tapping into new customer segments. The new customer segments could be the fair trade products where documentation is required as proof. This proof could be available from all downstream actors.

6.7 Limitations of the proof of concept & architecture.

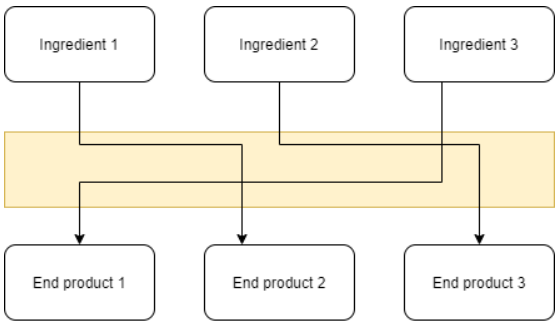
- The first limitation regards the designed architecture that assumes correct internal traceability. However, the quality of traceability for the whole supply chain is a combination of the internal and external traceability. If one actor does not have proper internal traceability, full supply chain traceability is almost impossible. Currently, there are two cases in which traceability is nearly impossible. These have been illustrated in figure 21.

    1. The first case is when multiple batches are combined into one large batch and become indistinguishable. This happens in the flour production when grain of multiple farmers is combined in a single batch of flour. If the flour is contaminated, it is nearly impossible to determine whose grain was contaminated. This is shown in figure 21a.
    2. The second case is when an actor has a lot of consignments arriving and leaving. When minimal or no internal data is captured, traceability is almost impossible. This is graphically illustrated in figure 21b. Without the internal traceability (yellow bar), it is almost impossible to determine which ingredient batch was used in an end product batch.

    Traceability of ingredients to end products is only possible if the producers share the internal information that links the input products with the output products. Therefore, a fraudulent actor that forges data from the beginning can obstruct traceability. Without the internal information, upstream and downstream actors can combine their data, but can only estimate which ingredients went into the sold product. However, once an actor shares (correct) pedigree URIs, the internal traceability of the second case is ensured. This covers the organic food scandal in Italy, where fraudsters made short-lived companies to obstruct traceability (Flari et al., 2014). Once the fraudsters have shared correct pedigree URIs, the trade flow can be reconstructed.



(a) This figure shows the combination of three batches of wheat intro one batch of flour. If the flour is contaminated, it is no longer possible to determine which wheat batch was contaminated



(b) This figure represents a production process that uses three ingredients and three possible end products. If the internal traceability is absent (yellow box), traceability becomes almost impossible.

Figure 21: This figure shows the importance of internal traceability.

- One of the goals of this research was to implement ontology based access control(OBAC) in the Linked pedigrees architecture. Due to the current state of Solid, this could not be proven to work. This affected both the architecture and the POC.

  Solid was said to be capable of defining access policies on the level of individual triples (Sambra et al., 2016). However, these were individual files (e.g. turtle files) when accessing data in the current solution. There were two reasons why OBAC did not work: first, OBAC required data requests to refer to the metadata. While in the artefact, data requests happened by direct request for a file at an endpoint, and thus, no metadata is included in the request. The second reason was that the graph of the data and the request should be compared. This could not be done, because requests do not include the graph of the data.

  Although OBAC could not be implemented according to the four requirements presented by the writers, the high-level idea of OBAC, to determine access policies based on the structure of the data was achieved. This can be observed in figure 29, where an actor defines a graph of data that can be accessed.

- The proof of concept does not differentiate between users of the application. This means that every employee can access all parts of the application. However, employees should have different rights according to their functions. E.g. the logistics employees only have access to the logistics information, while production employees must fill in the production steps taken and management could grant access to supply chain partners. The ideas used in pattern based access control could be used to improve this (Werbrouck et al., 2020).

7 Discussion & conclusions

In this section, the conclusions of the thesis are presented. The first subsection provides the answers to the research questions. Then generalisations towards other domains are made. After this, the limitations are identified. The thesis ends with future work.

The designed architecture is a possible solution for the lack of traceability in the agri-food supply chain. The goal of this thesis was to design a system that addresses the lack of traceability in the agri-food supply chain in a manner that conforms to the stakeholders' concerns. Existing traceability systems did not address the stakeholders' concerns regarding the safety of their data sufficiently. To prevent overlooked concerns, the concerns and barriers to adoption were identified. The concerns and barriers that had been identified, were translated to requirements for a traceability system. Then five possible approaches to develop a traceability system were analysed. Solid has the most promising requirement satisfaction as approach. Linked pedigrees was extended to work with Solid to create traceability combined with data access control. Moreover, Solid was demonstrated in a business to business context rather than the original social media context. An attempt was made to implement Ontology Based Access Control to create fine grained access control. Technological shortcomings in a component underlying the proof of concept, unrelated to the concept of OBAC or Linked pedigrees, impeded its proper implementation. Despite this setback, the proof of concept has demonstrated the fine-grained access control of OBAC by emulation. Finally, the proof of concept of the architecture has demonstrated the ability to find origins and destinations of products as well as supplementary data of the production process. This was achieved by queries to the different Solid PODs containing the traceability data. Moreover, the system is able to recover the origins of products when an actor revoked access to their data. This is achieved by an external backup of the data to find the suppliers of the actor.

7.1 research questions

7.1.1 RQ 1: What requirements should a traceability system satisfy in the agri-food supply chain?

11 requirements were found during this research. This research question is answered by the 11 requirements in chapter 3:

1. **Traceability - data** This requirement articulates the data that are required for traceability and required by law.
2. **Traceability - linked** This requirement articulates the demand for an explicit relation between data records of raw materials, ingredients, and (partial) food products.
3. **Traceability - query** This requirement articulates the demand to query through the traceability data. This is demanded because of the wide range of possible entrances of possible food safety incident clues.

4. **Traceability - backup** This requirement articulates the desire to have a backup mechanism. The backup mechanism should ensure the supply chain of a product can be reconstructed, not the data. This is required because fraudulent actors attempt to hamper traceability (Flari et al., 2014).

5. **Access control** This requirement articulates the demand for fine grained access control. This originates from the supply chain actors who are concerned about the safety of their commercial confidential data.

6. **Value proposition - compelling feature** This requirement articulates the desire for a compelling feature. This originates in the need for the entire supply chain to join a traceability system before it realises value. The compelling feature should ensure an actor can realise value on its own.

7. **Value proposition - data reuse** This requirement articulates the desire to reuse data from existing (legacy) systems. This is because stakeholders are concerned that traceability systems add to the workload of employees.

8. **Value proposition - easy to use** This requirement articulates the desire for traceability systems to be easy to use. This originates from previous systems that were perceived too difficult for end users.

9. **Trusted software supplier - supplier reputation** This requirement articulates the demand for a software supplier that can be trusted by the agri-food supply chain and has a good reputation. This requirement originates from the concerns of the stakeholders regarding the data safety.

10. **Trusted software supplier - reliable system** This requirement articulates the demand for a system that functions properly and continue to function properly. Additionally, the system should not generate a vendor lock-in.

11. **Trusted software supplier - governance** This requirement articulates the demand for a neutral governance body for the data, standards used and enterprise application integration. This requirement originates from the demands of the supply chain actors.

7.1.2 RQ 2: What are the possible treatments that satisfy the requirements?

This research question is answered in two categories: applications and approaches.

- **Applications** None of the available applications satisfied all the requirements. In the search for potential applications, five traceability systems were analysed. All systems lacked satisfaction of the access control requirement. However, access control is considered easiest to realise in the Linked pedigrees application. Therefore only Linked pedigrees is considered most promising.

- **Approaches** Solid was found the approach that satisfied the most requirements. Of the four approaches, none satisfied all the requirements. Solid and blockchains came closest, both had neutral requirements. In comparison, Solid satisfied more requirements than blockchains (9 vs. 5). Therefore, Solid was found the most promising approach. The other approaches (IPFS and IDS) were incapable of satisfying some requirements by their design, making them less suitable to use for a traceability system.

7.1.3 RQ3: What are the advantages and disadvantages of the available treatments?

The available approaches to be used as treatments have various advantages and disadvantages. These are listed in table 16 below. Additionally, the advantages and disadvantages of Linked pedigrees and ontology based access control (OBAC) over current access control frameworks are given.

Linked pedigrees	Advantage: Enables traceability while the data owner remains in control of their data. Advantage: A backup mechanism with minimal infringements on the data confidentiality. Disadvantage: Missing access control and compelling feature requirements.
Solid	Advantage: Data owner is in control of their data. Advantage: Available libraries for quick application development. Disadvantage: Not designed for a backup mechanism.
IDS	Advantage: Data owner is in control of their data. Disadvantage: Complicated environment to find new partners.
Blockchains	Advantage: Immutability of data. Advantage: Hype of blockchains. Disadvantage: Expensive data storage. Disadvantage: Difficult to remain in control over the data on the blockchain.
IPFS	Advantage: Fast upload and download speed. Disadvantage: Impossible to create access control.
OBAC	Advantage: Efficient way of managing access to similar resources. Advantage: Ability to allow access to specific resources. Disadvantage: Potential role explosion due to the role based background.

Table 16: Advantages and disadvantages of the found treatments.

One of the things that stood out, from the available application treatments was the choice of most systems for a blockchain (TE-Food, Food trust, and SeafoodIQ). Particularly, because one of the conclusions drawn from research question 2, was that better approaches to built traceability systems on were available. Particularly, as Food trust implies, that data on a blockchain can still be sold to or accessed by third parties out of control of the supply chain actors.

7.1.4 RQ4: What is an architecture that would satisfy the requirements for a traceability system in the agri-food supply chain?

The designed architecture that satisfies the requirements is presented in figure 22. The Linked pedigrees architecture was extended and implemented on the Solid ecosystem. Therefore, figure 22 has similar elements as described in the literature (Brewster et al., 2018; Sambra et al., 2016). The architecture has three key characteristics to satisfy the requirements: distributed but linked pedigrees, data separated from applications, and the integrated linked pedigrees store.

The distributed but linked pedigrees enable traceability while granting the supply chain actors access control over their traceability data. Moreover, actors that can control who has access to their data, are able to sell their data. The ability to sell data satisfies the value proposition - compelling feature requirement.

The separation of data and application enables an environment that requires application providers to built solutions that satisfy stakeholder goals and concerns. In that way, actors can choose an application and application provider which they trust and suits them. Moreover, the separated data and applications enable actors to switch to other application or POD providers when their current providers cannot be trusted.

The integrated linked pedigrees store realises the traceability - backup requirement. This is achieved in a stakeholder friendly manner by capturing the links between products, but not the data of the products itself.

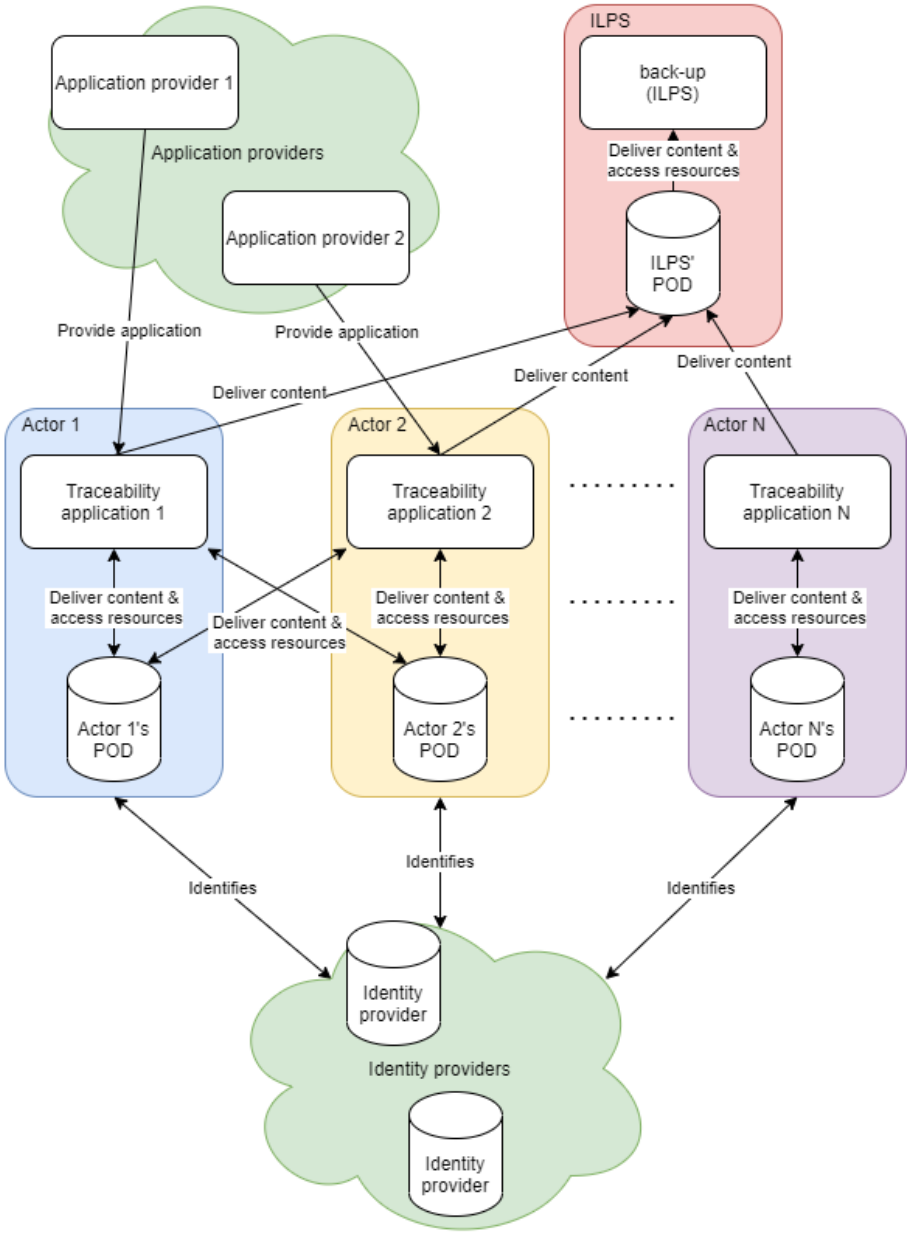


Figure 22: Overview of the designed architecture. This is the same architecture as presented in figure 7 and explained in section 5. Shows two supply chain actors(blue & yellow rectangles) that use a Linked pedigrees application from different application providers.

7.1.5 RQ5: How well does the architecture handle simulated food security incidents from the past?

The architecture was validated (section 6) capable of traceability in a model of the ready-made lasagne supply chain. Next to that, it demonstrated the potential to generate traceability in 3-4 seconds. Food trust can generate traceability in 1 second. Although 3-4 seconds is slower than food trust, it is a faster method than currently possible with paper-based traceability. Moreover, the validation demonstrated that the architecture satisfied the traceability requirements. The traceability - data requirement was satisfied by capturing production and logistics data in a pedigree. The traceability - linked requirement was satisfied by sharing the pedigree URIs with upstream and downstream actors. The combination of the pedigree URIs and the Solid enabled the realisation of the traceability - query requirement.

Moreover, the artefact is validated capable of handling with disappearing actors. This was simulated by removing access to the data of an actor. To re-find the origins of ingredients, the ILPS connected the known links (pedigree URIs) between products. However, it was not validated what happens when multiple actors disappear. This could create a problem in finding the root cause. In the validation, only one actor was assumed to disappear which raised a flag for investigation. When multiple actors disappear at the same time, such a clear flag is no longer present.

The artefact was validated to satisfy all other requirements. The artefact was validated able to create fine grained access control by emulating OBAC. The value proposition - compelling feature was satisfied by the ability to sell traceability data. The artefact realised this by the ability to specify persons who have access to the data. The value proposition - data reuse was satisfied by the ability to import data into the artefact. The value proposition easy to use was satisfied by the design of the Linked pedigrees architecture, which had a design to automate the process of capturing traceability data. The trusted software supplier requirements are satisfied by the Solid ecosystem, that enables multiple applications to be used, with a minimal vendor lock-in. This ensures that software suppliers that cannot be trusted will receive minimal uptake of their system. Therefore it is concluded that the artefact is capable of traceability, satisfies the requirements and can handle a simulated food security incident.

7.1.6 Contributions

This research has made the following contributions:

- Identified a list of requirements for traceability systems. Future designers of traceability can rely on the identified requirements to built a traceability system in line with stakeholder concerns. Because the requirements are based on needs of supply chain actors and barriers to adoption of previous traceability systems, implementing all requirements will lower the perceived barriers to adoption for stakeholders.

- Designed an architecture that is in line with stakeholder concerns and that can increase food safety. Moreover, The architecture was demonstrated in a proof of concept to realise traceability. This also demonstrated the possibility to implement Linked pedigrees.
- Finally, This architecture demonstrates new methods to realise traceability in a decentralised manner. It demonstrates Solid outside the original social media context and the realisation of traceability without a blockchain.

7.2 Applicability to other domains

The designed artefact should work in other domains where traceability is desired as well. This is because similar barriers to adoption (section 2.3) could be present. For example, the ROI of traceability systems relies on estimated benefits quantified by other means than direct income, like less losses due to an adequate response to product quality incidents. The realisation of these benefits relies on every supply chain actor participating. Next to that, the production data are of similar sensitivity resulting in similar distrust in sharing the data with other supply chain actors.

Although the necessity for traceability is prominent in food safety incidents, other domains could benefit from traceability as well. For example, the furniture industries would use the artefact to create proof of sustainable wood rather than tropical rainforest wood. Another example is chip production. Similar to the agri-food supply chain, when a supplier delivers faulty chips (comparable to contaminated food), end products like computers and smartphones do not function properly. Finding the faulty chip producer quickly should save production of faulty devices and negative branding. Additionally, precise recalls of defective products would be possible. For these use-cases, the artefact still realises benefits.

7.3 Limitations

One limitation of the architecture is that traceability data can be changed. This can be improved by immutable data. The Backup at the ILPS ensures that the pedigree URIs cannot be erased. However, the backup at the ILPS does not store the traceability data, only the links between pedigree URIs. Therefore, the ILPS cannot retrieve lost or changed traceability data, only recreate the chain of products. Moreover, the ILPS cannot validate if the data are correct. Therefore fraudsters can change or remove the data to hide the root cause of a food safety incident. A partial solution for this can be the use of hashing algorithms. Similar to IPFS, blockchains and nanopublications (Werbrouck et al., 2020), including the hash of a pedigrees' content into the URI. This allows validation of the content. However, this does not address the removal of traceability data.

7.4 Recommendations

Traceability software providers should consider capturing traceability data in the form of a pedigree. Then, actors should only share the access point of the data. In this way, actors can access upstream and downstream traceability data, while being in control who has access to it.

Future enterprise architects should consider architectures that separate the data from the application, like Solid does. Such systems allow control over company-owned data, easy application switching and data reuse, and application interoperability. Decoupling applications from the data would provide these benefits for organisations. Especially in organisations that have a large software landscape with interoperability problems. The ability to work with multiple applications on the same data (Sambra et al., 2016), can create synergy between applications in the application landscape. But, this requires semantic interoperability of the data. Moreover, decoupling the data from the application could lower the number of legacy systems. This is because organisations can switch between applications (Sambra et al., 2016) without transferring the data of the old system to the new system. Moreover, a gradual transition would be possible, so that employees can learn or test the new application while the main body of work still happens on the legacy system. However, this relies on the availability and interoperability of applications that decouple applications from the data.

7.5 Future work

Although the artefact was demonstrated capable of traceability, there are open questions, which have not been answered by this thesis.

- The first open question is how to implement internet of things (IoT) devices into the architecture to create (automatic) traceability. The automated traceability with IoT devises is desired because it would reduce the perceived effort to create traceability. As described in the limitations, traceability is depended on the quality of the internal traceability. Some options to create internal traceability include: integration with commercial of the shelf software and improving the architecture with IoT connections to automatically capture traceability data. Therefore, future research could investigate integration of IoT devices with the architecture, or with application-data decoupled architectures. Some work has been done in implementing IoT devices into the Solid architecture (Bader & Maleshkova, 2020). However, their focus lies on interaction between IoT devices and protocols used in communication between IoT devices.

- A potential step for future research is the development of detection algorithms for potential food safety incidents. In the current state of the artefact, the traceability data can be captured, and cross actor queries can be programmed. In an ideal situation, this would be monitored automatically. Moreover, such detection algorithms could be used to test for valid traceability data. For example, the algorithms could detect fraudulent

behaviours in traders that say they bought more products from a farmer than he can reasonably produce. Such behaviour was seen in the organic food crisis in Italy (Flari et al., 2014), and should be detectable.

- A final direction for future research is the incorporation of contracts for data access that agree on rules for replication and sharing the data to others. The artefact allows this by a manual process at the moment a contract is accepted over e-mail or in person. But, this could also be automated in a manner similar to IDS, where rules on data reuse have to be accepted before data access.

References

aelf. (2019). *The Dapper Weekly — Ep 14 - IBM Food Trust Review (A Billion Dollar App?).* Retrieved from https://youtu.be/sSu4Zymu2n8

Bader, S. R., & Maleshkova, M. (2020). SOLIOT — Decentralized Data Control and Interactions for IoT. , *12*(105), 1–31. doi: 10.3390/fi12060105

Bánáti, D. (2014). European perspectives of food safety. *Journal of the Science of Food and Agriculture*, *94*(10), 1941–1946. doi: 10.1002/jsfa.6611

Bosona, T., & Gebresenbet, G. (2013). Food traceability as an integral part of logistics management in food and agricultural supply chain. *Food Control*, *33*(1), 32–48. Retrieved from http://dx.doi.org/10.1016/j.foodcont.2013.02.004 doi: 10.1016/j.foodcont.2013.02.004

Brewster, C., Nouwt, B., Raaijmakers, S., & Verhoosel, J. (2019). Ontology-based access control for FAIR Data. *Data Intelligence*(special issue DI-2019-0006).

Brewster, C., Seepers, R., & all WP Participants. (2018). Ensuring the Integrity of the European Food Chain: Overview and Showcases demo. *UE Project FP7-KBBE 613688*, *613688*(613688). Retrieved from https://secure.fera.defra.gov.uk/foodintegrity/index.cfm

Buchholz, U., Bernard, H., Werber, D., Böhmer, M. M., Remschmidt, C., Wilking, H., ... Kühne, M. (2011). German Outbreak of Escherichia coli O104:H4 Associated with Sprouts. *New England Journal of Medicine*, *365*(19), 1763–1770. Retrieved from https://doi.org/10.1056/NEJMoa1106482 doi: 10.1056/NEJMoa1106482

CBS. (n.d.). *Bevolkingsteller.* Retrieved 2020-06-26, from https://www.cbs.nl/nl-nl/visualisaties/bevolkingsteller

Content blockchain. (n.d.). *Exploding Costs of storing information on a Blockchain.* Retrieved 2020-03-30, from https://content-blockchain.org/newsarchive/2017/07/20/exploding-costs-of-storing-data-on-a-blockchain/

Czinkota, M., Kaufmann, H. R., & Basile, G. (2014). The relationship between legitimacy, reputation, sustainability and branding for companies and their supply chains. *Industrial Marketing Management*, *43*(1), 91–101. Retrieved from http://dx.doi.org/10.1016/j.indmarman.2013.10.005 doi: 10.1016/j.indmarman.2013.10.005

Dediu, L., Moga, L. M., & Cristea, V. (2016). The barriers for the adoption of traceability systems by Romanian fish farms. *AACL Bioflux*, *9*(6), 1323–1330.

Dignan, L. (2018). *IBM Food Trust blockchain network available, Carrefour joins retailer roster.* Retrieved 2020-03-31, from https://www.zdnet.com/article/ibm-food-trust-blockchain-network-available-carrefour-joins-retailer-roster/

Duan, Y., Miao, M., Wang, R., Fu, Z., & Xu, M. (2017). A framework for the successful implementation of food traceability systems in China. *Information Society*, *33*(4), 226–242. doi: 10.1080/01972243.2017.1318325

Elliott, A., & Knight, S. (2010). Role Explosion: Acknowledging the Problem. *Software Engineering Research and Practice, WORLDCOMP*(October 1992), 349–355.

Everstine, K., Spink, J., & Kennedy, S. (2013). Economically motivated adulteration (EMA) of food: Common characteristics of EMA incidents. *Journal of Food Protection*, *76*(4), 723–735. doi: 10.4315/0362-028X.JFP-12-399

Feiten en cijfers. (n.d.). Retrieved 2020-06-26, from https://nieuws.ah.nl/feiten-en-cijfers/

Felicity, L. (2013a, 2). *Horsemeat scandal: the essential guide.* Retrieved 2020-02-15, from https://www.theguardian.com/uk/2013/feb/15/horsemeat-scandal-the-essential-guide

Felicity, L. (2013b, 2). *Horsemeat scandal: the essential guide.* Retrieved from https://www.theguardian.com/uk/2013/feb/15/horsemeat-scandal-the-essential-guide

Flari, V., Hussein, M., Maeder, R., Huber, B., Marvin, H., & Neslo, R. (2014). Ensuring the Integrity of the European food chain Report on analysis of historical cases of food fraud.

GMA, & Kearney, A. (2010). CONSUMER PRODUCT FRAUD: DETERRENCE AND DETECTION. Retrieved from https://www.gmaonline.org/downloads/research-and-reports/consumerproductfraud.pdf

Google Search Statistics. (n.d.). Retrieved 2020-06-26, from https://www.internetlivestats.com/google-search-statistics/%0A

GS1. (n.d.). *EPCIS and Core Business Vocabulary (CBV).* Retrieved 2020-07-31, from https://www.gs1.org/standards/epcis

Hall, D. (2010). Food with a visible face: traceability and the public promotion of private governance in the food system. *GeoForum*(41), 826–835.

Hardt, M. J., Flett, K., & Howell, C. J. (2017). Current Barriers to Large-scale Interoperability of Traceability Technology in the Seafood Sector. *Journal of Food Science*, *82*, A3-A12. doi: 10.1111/1750-3841.13796

Hittle, B., & Leonard, K. M. (2011). Decision making in advance of a supply chain crisis. *Management Decision*, *49*(7), 1182–1193. doi: 10.1108/00251741111151208

IBM. (n.d.-a). *From bean to the brew on the blockchain.* Retrieved 2020-03-31, from https://www.ibm.com/thought-leadership/coffee/

IBM. (n.d.-b). *IBM Food Trust.* Retrieved 2020-03-31, from https://www.ibm.com/blockchain/solutions/food-trust

Kantar Worldpanel. (2013). *Grocery Market Share UK - First Data Since Horsemeat Scandal.* Retrieved 2019-09-16, from http://www.kantarworldpanel.com/global/News/Grocery-Market-Share-UK-First-Data-Since-Horsemeat-Scandal

Kwon, H.-k., & Seo, K.-k. (2013). Application of Value-based Adoption Model to Analyze SaaS Adoption Behavior in Korean B2B Cloud Market. *International Journal of Advancements in Computing Technology*, *5*(12), 368–373.

Labs, P. (n.d.). *IPFS Documentation.* Retrieved 2020-03-26, from https://docs.ipfs.io/

Lee, Y., & Kozar, K. A. (2008). An empirical investigation of anti-spyware software adoption: A multitheoretical perspective. *Information and Management*, *45*(2), 109–119. doi: 10.1016/j.im.2008.01.002

Manning, L., & Soon, J. M. (2014). Developing systems to control food adulteration. *Food Policy*, *49*(P1), 23–32. doi: 10.1016/j.foodpol.2014.06.005

Manning, L., & Soon, J. M. (2016). Food Safety, Food Fraud, and Food Defense: A Fast Evolving Literature. *Journal of Food Science*, *81*(4), R823-R834. doi: 10.1111/1750-3841.13256

Mansour, E., Sambra, A. V., Hawke, S., Zereba, M., Capadisli, S., Ghanem, A., ... Berners-Lee, T. (2016). A Demonstration of the Solid Platform for Social Web Applications. , 223–226. doi: 10.1145/2872518.2890529

Minnens, F., Sioen, I., van de Burg, F., Luijckx, N. l., & Verbeke, W. (2018). Ensuring the Integrity of the European food chain: report on stakeholder attitudes towards information sharing along food supply chain.

Morris, N. (2019). *Uploading data to IBM's Food Trust blockchain is free.* Retrieved 2020-03-31, from `https://www.ledgerinsights.com/ibm-food-trust-blockchain-cost-food-traceability/`

Nnamdi O. Madichie, & Yamoah, F. A. (2017). Revisiting the European Horsemeat Scandal: The Role of Power Asymmetry in the Food Supply Chain Crisis. *Thunderbird International Business Review*, *59*(6), 663–675. doi: 10.1002/tie

Novum. (2007, 10). *Hema-worst wordt bij Unox gemaakt.* Amsterdam. Retrieved from `https://www.nu.nl/economie/1286516/hema-worst-wordt-bij-unox-gemaakt.html`

O'Connor, J. (2016). *Prioritizing Your User Stories with the MoSCoW Method.* Retrieved 2020-04-29, from `https://medium.com/@sax1johno/prioritizing-your-user-stories-with-the-moscow-method-8bf42d427da6`

Olsen, P., & Borit, M. (2013). How to define traceability. *Trends in Food Science and Technology*, *29*(2), 142–150. Retrieved from `http://dx.doi.org/10.1016/j.tifs.2012.10.003` doi: 10.1016/j.tifs.2012.10.003

OneDrive-abonnementen vergelijken. (n.d.). Retrieved 2020-06-26, from `https://www.microsoft.com/nl-nl/microsoft-365/onedrive/compare-onedrive-plans?market=nl&activetab=tab:primaryr2`

Otto, B., Steinbuß, S., Teuscher, A., & Lohmann, S. (2019). International Data Space Reference Architecture Model Version 3.0. (April), 118.

Panetta, K. (2019). *Gartner Top 10 Strategic Technology Trends for 2020.* Retrieved 2020-05-05, from `https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2020/`

Randrup, M., Storøy, J., Lievonen, S., Margeirsson, S., Árnason, S. V., Ólavsstovu, D. , ... Frederiksen, M. T. (2008). Simulated recalls of fish products in five Nordic countries. *Food Control*, *19*(11), 1064–1069. doi: 10.1016/j.foodcont.2007.11.005

Robertson, J., & Robertson, S. (2007). *Volere Requirements Specification Template edition 13.* the Atlantic Systems Guild.

Rosian, A., & Pustjens, M. (2019). *Online boodschappen doen heeft nauwelijks impact op de supermarkt.* Retrieved 2020-06-26, from https://www2.colliers.com/nl-nl/research/20190726supermarkten

Sambra, A. V., Mansour, E., Hawke, S., Zereba, M., Greco, N., Ghanem, A., ... Berners-Lee, T. (2016). SoLiD: A Platform for Decentralized Social Applications Based on Linked Data. *Technical report, MIT CSAIL & Qatar Computing Research Institute.*. Retrieved from https://diasporafoundation.org

SeafoodIQ. (2018a). *NEW TOOLBOX FOR BETTER TOMORROW.* Retrieved 2020-04-03, from https://seafoodiq.com/solutions/

SeafoodIQ. (2018b). *SUPPLY CHAIN 4.0 / Seafood IQs corporate video.* Retrieved 2020-04-03, from https://www.youtube.com/watch?v=k3jG266sVJM&feature=emb_logo

SeafoodTrace: Intelligent Traceability Platform enabling full transparency in the Seafood supply chain. (2018). Retrieved 2020-04-03, from https://cordis.europa.eu/project/id/816070

Seethamraju, R. (2015). Adoption of Software as a Service (SaaS) Enterprise Resource Planning (ERP) Systems in Small and Medium Sized Enterprises (SMEs). *Information Systems Frontiers*, *17*(3), 475–492. doi: 10.1007/s10796-014-9506-5

Solanki, M., & Brewster, C. (2014). Enhancing visibility in EPCIS governing agri-food supply chains via linked pedigrees. *International Journal on Semantic Web and Information Systems*, *10*(3), 45–73. doi: 10.4018/IJSWIS.2014070102

Spool, J. M. (2019). *Understanding the Kano Model — A Tool for Sophisticated Designers.* Retrieved 2020-04-29, from https://medium.com/@jmspool/understanding-the-kano-model-a-tool-for-sophisticated-designers-d91d092ad885

Steichen, M., Fiz, B., Norvill, R., Shbair, W., & State, R. (2018). Blockchain-Based, Decentralized Access Control for IPFS. *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 1349–1354. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8726562 doi: 10.1109/Cybermatics

Storoy, J., Thakur, M., & Olsen, P. (2013). The TraceFood Framework - Principles and guidelines for implementing traceability in food value chains. *Journal of Food Engineering*, *115*(1), 41–48. doi: 10.1016/j.jfoodeng.2012.09.018

TE-FOOD. (n.d.). *Solution.* Retrieved 2020-03-31, from https://tefoodint.com/solution.html

TE-FOOD. (2018). *Durian traceability on blockchain - TE-FOOD pilot in Vietnam.* Retrieved 2020-03-31, from https://www.youtube.com/watch?v=5I0zexC-Mw4

TE-FOOD. (2019). *Food Traceability Trends to watch in 2019.* Retrieved 2020-

03-31, from https://medium.com/te-food/food-traceability-trends-to-watch-in-2019-179a00b3b625

TRACES: TRAde Control and Expert System. (n.d.). Retrieved 2020-05-04, from https://ec.europa.eu/food/animals/traces_en

van Ruth, S. M., Huisman, W., & Luning, P. A. (2017). Food fraud vulnerability and its key factors. *Trends in Food Science and Technology*, *67*(June), 70–75. doi: 10.1016/j.tifs.2017.06.017

Welt, B., & Blanchfield, J. R. (2012). Food Traceability. *Handbook of Food Analysis, Third Edition - Two Volume Set*(March), 265–272. doi: 10.1201/b18668-16

Werbrouck, J., Taelman, R., Verborgh, R., Pauwels, P., Beetz, J., & Mannens, E. (2020). Pattern-based access control in a decentralised collaboration environment. (June), 1–14.

Wieringa, R. J. (2014). *Design science methodology for information systems and software engineering.* Springer.

Wiseman, L., Sanderson, J., & Robb, L. (2018). Rethinking Ag Data Ownership. *Farm Policy Journal*, *15*(1), 71–77.

Writing a Solid application. (n.d.). Retrieved 2020-06-24, from https://solidproject.org/for-developers/apps/first-app

Wu, W. W. (2011). Developing an explorative model for SaaS adoption. *Expert Systems with Applications*, *38*(12), 15057–15064. Retrieved from http://dx.doi.org/10.1016/j.eswa.2011.05.039 doi: 10.1016/j.eswa.2011.05.039

Yang, Z., Sun, J., Zhang, Y., & Wang, Y. (2015). Understanding SaaS adoption from the perspective of organizational users: A tripod readiness model. *Computers in Human Behavior*, *45*, 254–264. Retrieved from http://dx.doi.org/10.1016/j.chb.2014.12.022 doi: 10.1016/j.chb.2014.12.022

A Proof of concept

Because some aspects provide limited new scientific insights they were left out of the proof of concept. E.g. the import of data has been left out, as this feature is present in many of today's web applications. Similarly, the connection of IoT devices has been left out as there already is an abundance of research on IoT devices. The ability to show the pedigree chain downstream has also been left out as this is functionally similar to the upstream version. To save time domain model made with the OntoPedigree ontology has been simplified to a minimal. The ability to request additional data has been left out as this could also happen with the combination of a telephone or email and the *partner list* interface.

The proof of concept was implemented using a nodeJS server[20]. It is built as an extension on the example solid application (*Writing a Solid application*, n.d.). The Solid community server has been used as POD provider [21]. The solid community server is used to store pedigrees and additional data of the actors in

[20] https://nodejs.org/en/

[21] https://solid.community/

the validation scenario (section 6). An additional POD has been made for the implementation of the ILPS.

In figure 23 till figure 32, an impression of the proof of concept can be found. The top of the application has an orientation bar containing buttons that open the interfaces described in the technology architecture. The orientation bar is always present. The first interface is the overview interface, shown in figure 15. This interface shows a list of pedigrees. The edit button will open a popup (figure 24) where additional data of the pedigree can be viewed and entered. Additionally, the pop-up window contains an *update ILPS* button. This button is used to create the backup traceability and realise the *Send pedigree URI (B and A)* of the communication protocol in figure 14. Additionally, this popup has a *browse...* button. This is used to import data from a JSON object into the pedigree. The next interface, *new pedigree*, shown in figure 25, is used to create a new pedigree. In this interface three fields have to be filled: the pedigree name, which is a name the actors can give to a pedigree; The pedigree URI extension grants the opportunity to create a pedigree URI of their choice[22]; The product or service defined by the good relations ontology[23]. Figure 26, shows the *partner list* interface. This interface is used to create new partners, assigning them to an access control group and grant them access to a pedigree. The top half is used to create new partners of an actor. The *partner name* is the legal business name of the partner[24]. The *partner card URI* is the URI of the partners' profile document used for authentication. On the bottom half of the partner list, is the list of partners. Each partner can be assigned to an access control group with the *assign to group* button. This opens 27 where an access control group can be chosen. Figure 28 shows the popup used to view all the pedigrees a partner has access to within the limits defined in the access control interface. The *access control* interface is shown in figure 29 and used to create access control groups and edit the permissions of these groups. The top half is used to create new access groups. By default, an access group does not receive any access rights. The *define policies* button is used to open a popup where the access rights can be edited. This is the popup shown in figure 29. This popup has a button for every triple in the OntoPedigree ontology. Actors can define the access for each triple by clicking the respective button. This will change the colour between red and blue. Blue means the group has access to the triple, red means that the group has no access to the triple. The *send URI request* and *Send URI response* interfaces are used in the realisation of linked traceability data and backup traceability. The *Send URI request* is shown in figure 30 and is used to send a request for a pedigree URI to an upstream actor. It requires an actor to fill in the profile card of the owner and the EPC of the product. The *Send URI response* interface is shown in figure 31 and is used by the upstream actor to send a response to the downstream player. The interface shows a list of received pedigree requests. By clicking the *connect URI* button, a popup

[22]Base of the URI will still be the storage location mentioned in the profile document, most likely the POD providers' part of the URI.

[23]http://purl.org/goodrelations/v1#ProductOrService

[24]http://purl.org/goodrelations/v1#legalName

will open where the matching pedigree can be selected. In the background, a message is sent to the ILPS to create the backup traceability. The *view pedigree chain* interface is used to view the traceability chain of a product and is shown in figure 32. The overview requires a pedigree URI to be entered. If the URI is valid, all data in this URI will be retrieved and displayed in a row. If triples containing the derivedFrom predicate are found, these URIs will be used in subsequent rows to retrieve additional data. In the case shown in figure 32, the ready-made meals actor does not have access to data at the pasta producer or the sauce maker. For downstream traceability, a similar interface is made, the only difference being the derivedFrom predicate has been changed with usedIn.

Figure 23: Orientation bar of the proof of concept. Shows the tabs available in the proof of concept

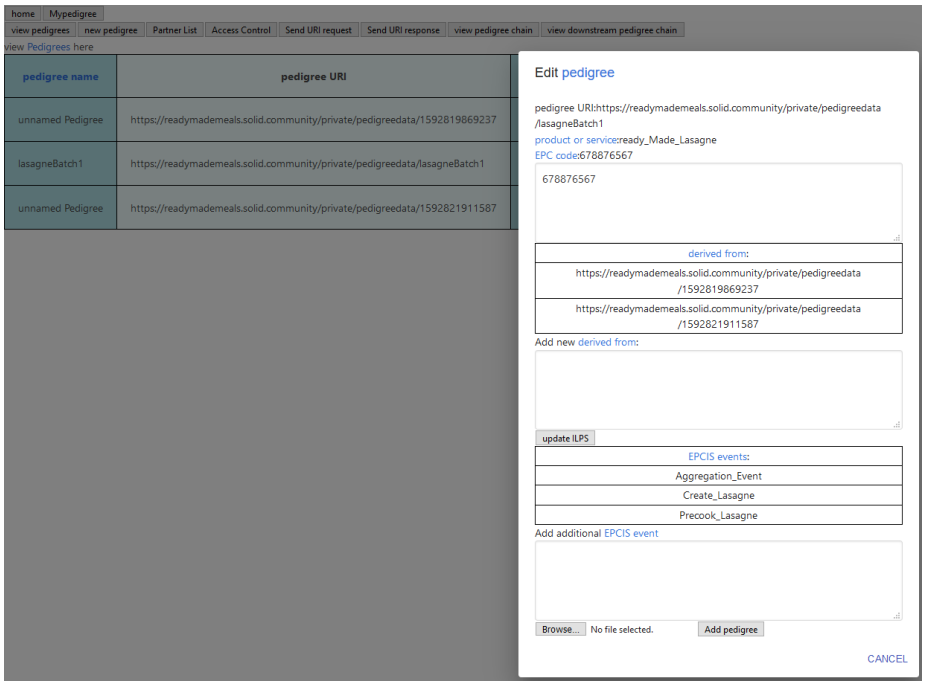


Figure 24: Popup used to edit and add data to a pedigree.

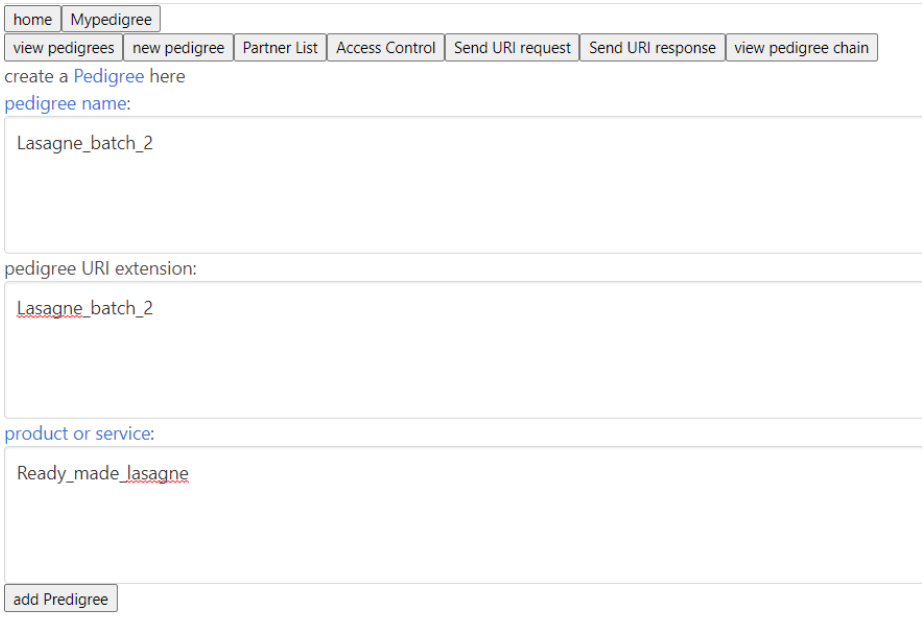


Figure 25: New pedigree interface. Used to create new pedigrees.

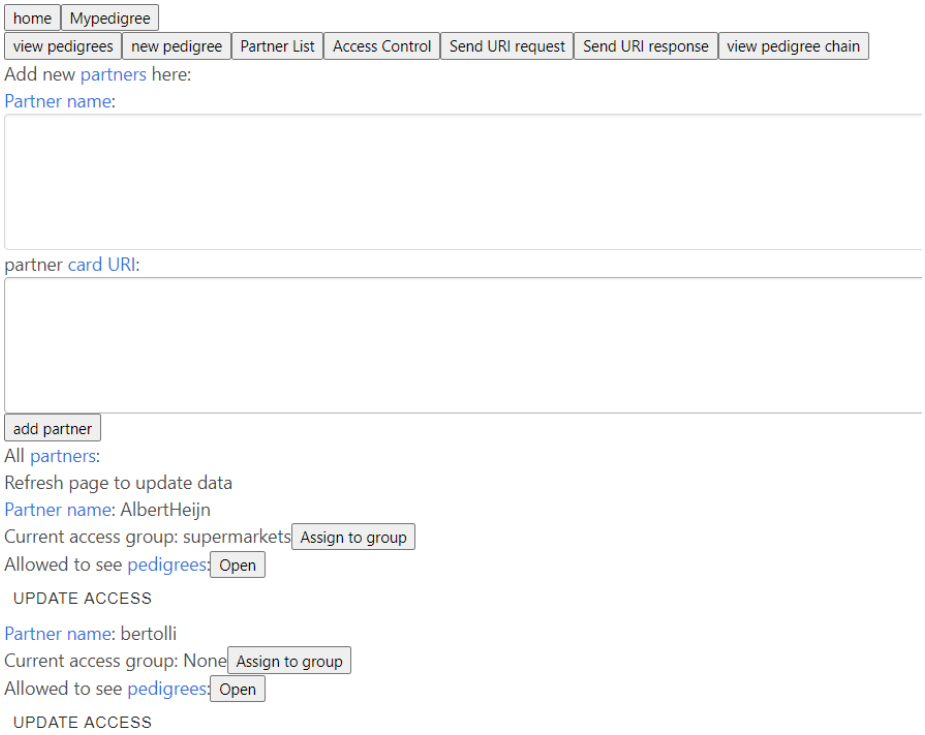


Figure 26: Partner List interface. used to create new partners and assign access rights to partners.

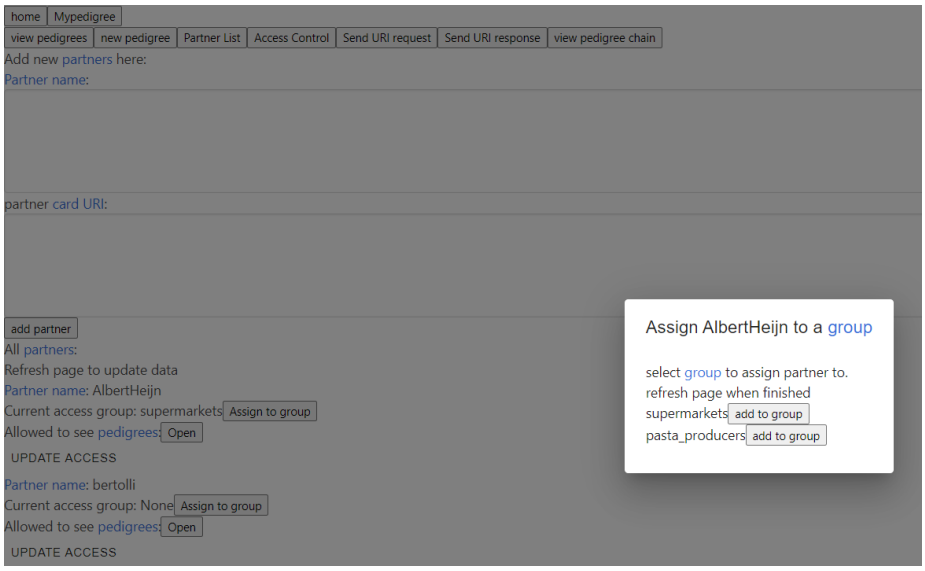


Figure 27: Popup used to assign partners to an access control group.

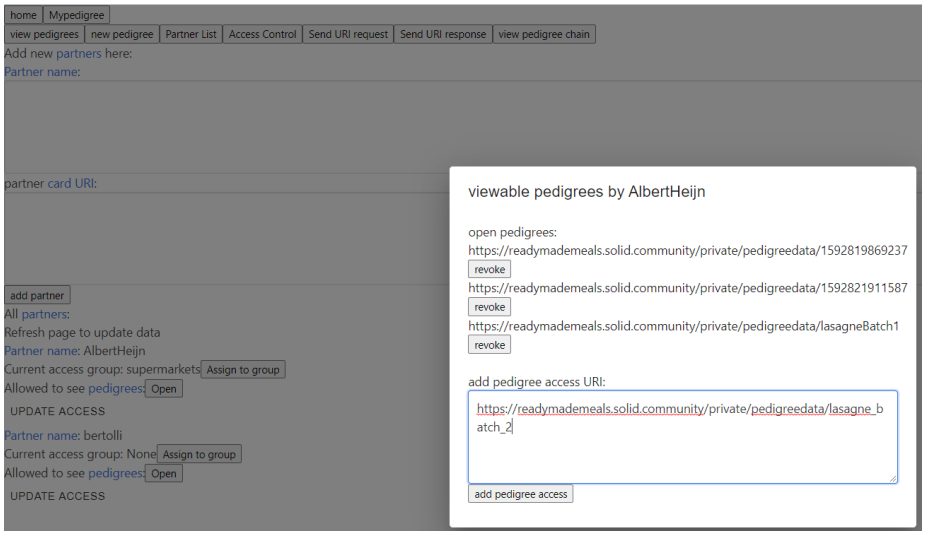


Figure 28: Popup used to grant partners access to a pedigree. Shows a list of pedigrees that are currently visible by the AlbertHeijn actor. Currently, they can view three pedigrees, and a fourth is being added.

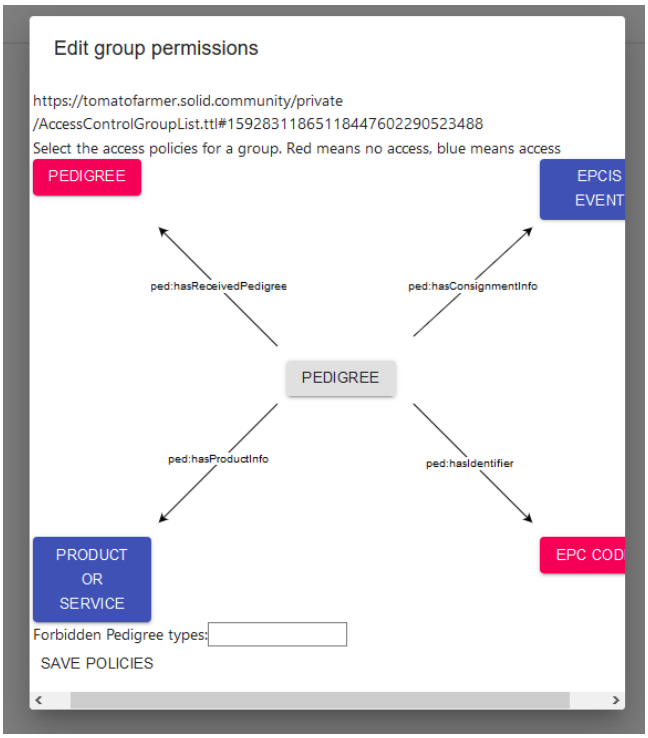


Figure 29: Access control popup. Used to define the policies for an access control group. Shows an interactive figure of the domain model used. Shows four possible objects in a pedigree for which access can be defined. The selected access control group has access to the product or service triples, and EPCIS event triples. On the bottom, an input field is present where product codes can be entered, that should not be viewable by this access group.

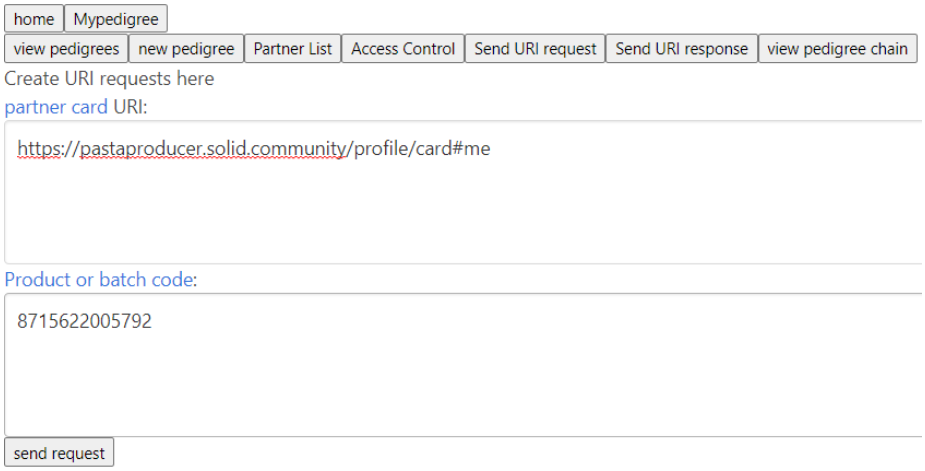


Figure 30: Send URI request interface. Used to create URI requests at upstream actors. Requires the card of the upstream actor as an endpoint to send the request to. The Product or batch code is used to identify the corresponding pedigree at the upstream actor.

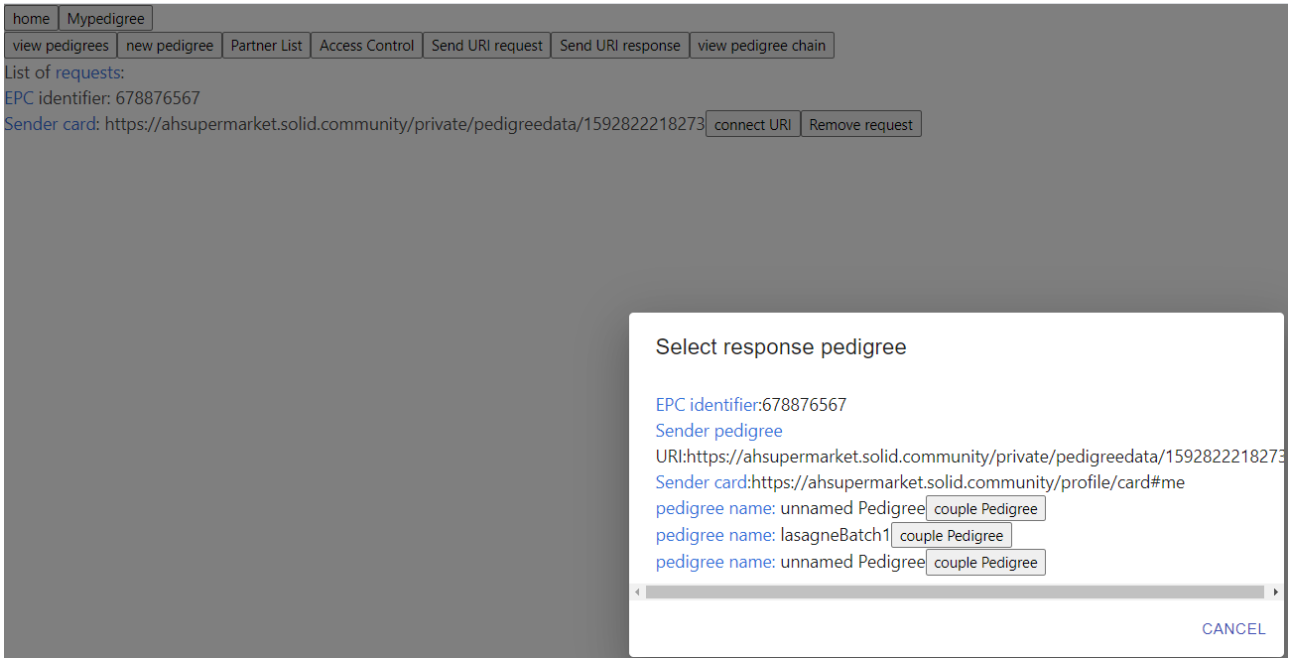


Figure 31: Send URI response interface, used to respond to the requests received, created in figure 30. The popup is used to select the corresponding pedigree matching the request.

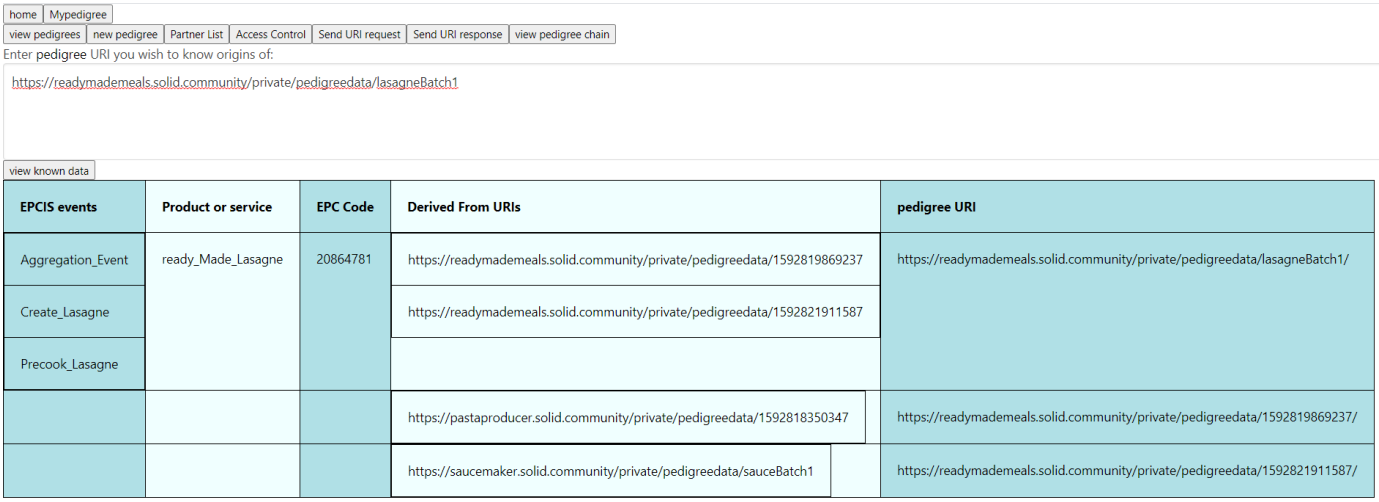


Figure 32: View pedigree chain interface. Used in finding data of upstream actors. Shows the data viewable by the ready-made meals actor. Shows the pedigree of the ready-made lasagne and two ingredients.

The ILPS has been implemented as a separate application. The ILPS has one interface, showing all the messages received from the actors in the supply chain. The interface of the ILPS is shown in figure 33. The ILPS has been made in an analogy of a double linked list. Each node(item in the list) contains a reference to the previous node, next node and a single pedigree URI. The columns in figure 33 contain the node, the next node, the previous node and the pedigree URI. The previous node corresponds to the derived from predicate and the next node corresponds to the used In predicate. E.g. The first row in figure 33 has a pedigree URI "https://draap.solid.community/private/pedigreedata/1592820713661". It also has a previousNode object. This previousNode object is the third row containing the pedigree: "https://cattlefarmer.solid.community/private/pedigreedata/betsi".

pedigree chain records at ILPS

fetch messages

Node number	PedigreeUri	Next node	previousNode
https://ilps.solid.community /pedigreeChain.ttl#1592919484539319746176148479 2	https://draap.solid.community/private/pedigreedata/1592820713661	https://ilps.solid.community /pedigreeChain.ttl#1592919487171520055828542182 6	https://ilps.solid.community /pedigreeChain.ttl#1592919484539684320140867693 3
https://ilps.solid.community /pedigreeChain.ttl#1592919487171520055828542182 6	https://beeftrader.solid.community/private/pedigreedata /1592821247026	https://ilps.solid.community /pedigreeChain.ttl#1592919497332715585062180461 6	https://ilps.solid.community /pedigreeChain.ttl#1592919484539319746176148479 2
https://ilps.solid.community /pedigreeChain.ttl#1592919484539684320140867693 3	https://cattlefarmer.solid.community/private/pedigreedata/betsi	https://ilps.solid.community /pedigreeChain.ttl#1592919484539319746176148479 2	
https://ilps.solid.community /pedigreeChain.ttl#1592919497332715585062180461 6	https://saucemaker.solid.community/private/pedigreedata /1592821489478		https://ilps.solid.community /pedigreeChain.ttl#1592919487171520055828542182 6
https://ilps.solid.community /pedigreeChain.ttl#1592919489842695975168938142 4	https://wheatmiller.solid.community/private/pedigreedata /1592814754688	https://ilps.solid.community /pedigreeChain.ttl#1592919494802722665931481041 1	https://ilps.solid.community /pedigreeChain.ttl#1592919489842913030056873583 9
https://ilps.solid.community /pedigreeChain.ttl#1592919494802722665931481041 1	https://pastaproducer.solid.community/private/pedigreedata /1592818350347	https://ilps.solid.community /pedigreeChain.ttl#1592919502292999881113637449 2	https://ilps.solid.community /pedigreeChain.ttl#1592919489842695975168938142 4
https://ilps.solid.community /pedigreeChain.ttl#1592919489842913030056873583 9	https://wheatfarmer.solid.community/private/pedigreedata /WheatBatch1	https://ilps.solid.community /pedigreeChain.ttl#1592919489842695975168938142 4	
https://ilps.solid.community /pedigreeChain.ttl#1592919492342070978199009393 62	https://readymademeals.solid.community/private/pedigreedata /lasagneBatch1	https://ilps.solid.community /pedigreeChain.ttl#1592919492342617084055196111 4	
https://ilps.solid.community /pedigreeChain.ttl#1592919492342617084055196111 4	https://ahsupermarket.solid.community/private/pedigreedata /1592822218273		https://ilps.solid.community /pedigreeChain.ttl#1592919492342070978199009393 62
https://ilps.solid.community /pedigreeChain.ttl#1592919502292999881113637449 2	https://readymademeals.solid.community/private/pedigreedata /1592819869237		https://ilps.solid.community /pedigreeChain.ttl#1592919494802722665931481041 1
https://ilps.solid.community /pedigreeChain.ttl#1592919499756546331229584079 6	https://saucemaker.solid.community/private/pedigreedata/sauceBatch1	https://ilps.solid.community /pedigreeChain.ttl#1592919499756755351303935452	

Figure 33: Overview of the data in the ILPS. Shows the data of the validation scenario.