University of Groningen
Faculty of Science and Engineering

v1.0.0

Software Architects:

Amey Bhole
Andrei Cusnir
Remco van Buijtenen
Stefan Evanghelides
Thorsten Rangnau

Project Supervisor:

Dan Chirtoaca

Course Lecturer:

Paris Avgeriou

November 2019

Authors

Table 1 displays the students who have contributed to the creation this document:

Name	ID	Student Number	Email address
Amey Bhole	ab	s3411427	a.bhole@student.rug.nl
Andrei Cusnir	ac	s3608662	a.cusnir@student.rug.nl
Remco van Buijtenen	rb	s2714086	r.m.van.buijtenen@student.rug.nl
Stefan Evanghelides	se	s2895323	s.evanghelides@student.rug.nl
Thorsten Rangnau	tr	s3570282	t.rangnau@student.rug.nl

Table 1: Information about authors

Change log

Table 2 lists all versions and revision details of this document.

Version	ID	Date	Description
0.1.0	se	08-09-2018	Added the title page, change log and included Andrei's research chapter 2.3 under Crop Farming subsection
	tr	08-09-2018	Create first draft document, wrote general part and aquaculture part of chapter 1, wrote chapter 2.1, 2.2, 2.3 (general part and aquaculture part), 2.4 and 2.8
	ab	08-09-2018	Animal Farming and added a paragraph to System context and Target audience related to the animal farming
	rb	08-09-2018	Chapter 3 - stakeholders, functional requirements, commercial requirements, non functional requirements, evolution requirements and risk assessment
	ac	08-09-2018	Chapter 1 crop protection and monitoring paragraph, Chapter 2 poultry paragraph, Chapter 2.4 target audience for crop farming
0.2.0	se	12-09-2018	Switched report from Word-style to LaTeX. Added Authors and Change log sections.
		15-09-2018	Contributed in the stakeholder interest ranking in section 4.2. Added Glossary section, and provided explanations for the relevant keywords. Updated section 3.4.
	tr	15-09-2018	Add Chapter 4 and first draft of it.
		16-09-2018	Create diagrams for Use Case 01, 02 and align corresponding section
	rb	15-09-2018	Updated sections 4.2, 4.4, 4.5, 4.6
	ab	16-09-2018	Added Introduction and updated sections 2, 3.1, 3.2, 3.3, 3.5, 4.7, 5.1
	ac	16-09-2018	Updated sections 3.7, 3.8 and 3.9 and fixed stakeholders during the meeting
0.3.0	ac	20-09-2018	Edited Use-Cases scenarios and fixed the content. Worked on Logo DEMO. Updated: Commercial non-functional requirements, Technical non-functional requirements and Evolution Requirements. Created SWOT Analysis table
	se	21-09-2018	Updated the Financial Model, added references in the footnote

		23-09-2018	Added change bars, removed outdated references, replaced future tense by present tense + other small grammar fixes
	tr	22-09-2018	Update Use Cases, create diagrams for business model, architectural vision, architectural high level design and design solution to minimize risks
		23-09-2018	Create domain model diagram, updating risks section and update high level design solution diagram, domain model subsection, add high level design decision Tables (1 - 5) to chapter 5
	ab	22-09-2018	Updated Business Rationale, Target Audience, Roadmap and Competitors
		23-09-2018	Updated Business Vision, Stakeholders, Product/Service description, Business model
	rb	22-09-2018	Added architectural vision (4.1)
		23-09-2018	Updated risk assessment (4.10), high level design diagram (5.3), high level design decisions 1 and 5.
0.4.0	ac	28-09-2018	System Architecture draft started, Initial models, Actors and Roles and Components and External Systems
	ac	29-09-2018	Redefine Risks and create new ones
	se, ab	29-09-2018	Updated stakeholder ranking, concern matrix, key drivers.
	tr	29-09-2018	5.1 assumptions
	rb	29-09-2018	4, 4.1 Architectural Vision; 4.8 Added a risk; 5.3 Decisions 5 and 6
0.5.0	ac	02-10-2018	Redefined Assumptions, removed the wrong ones, other fixes
	tr	02-10-2018	Add high-level requirements, change domain model and UC diagram
		06-10-2018	Created draft for chapter 8.1
		07-10-2018	Created draft for chapter 8.2
	ac	03-10-2018	Created draft for Hardware chapter
	ab	03-10-2018	Discussed the architectural vision diagram with Remco and updated the stakeholder text
	ab	07-10-2018	Updated the text in system architecture for 6.1 intial model and 6.2 elaborated model and updated and added diagrams for inital model and elaborated model. Added text for availability in key drivers
	rb	03-10-2018	4.1 Created architectural vision diagram
		06-10-2018	4.1 Updated architectural vision text; 5.3 Added decision 7, intro paragraph and decision mapping table 6.1.2 worked with Amey on initial model diagram
	se	03-10-2018	Refined concern matrix based on the feedback.
		07-10-2018	Updated risks table.
0.6.0	tr	09-10-2018	Replace wrong business model figure, fixed high-level requirements
		13-10-2018	Replace initial model diagram, replace use case diagram
		14-10-2018	Refactor Use Cases section, extend chapter 8.2 and create class diagram for farm component
	se	10-10-2018	Split the types of risks in separate tables. Populated them with more risks with the help of the team. Moved them to the appendix.
		14-10-2018	Added subsection 4.6.1 mapping FR's into the HLR's.

	ac	14-10-2018	6.3.2 Time to Market draft of the paragraph, 6.3.3 Cost and Return on Investment draft of the paragraph, 5.1 Assumptions rewrite 6th assumption
	ab	14-10-2018	Created Technology roadmap and updated section 5.2. Added section 6.3.1 and updated 6.1.2
	rb	10-10-2018	4.1 update architectural vision diagram
		14-10-2018	4.2 update stakeholder description
			4.9 update risk tables
			6.2 updated elaborated model diagram
			7.1 & 7.2 update paragraphs based on feedback
			8.1 added two references
0.7.0	ac	18-10-2018	Draft of Chapter 9 Architecture Evaluation
	ac	19-10-2018	Fix stake holders relevance to key drivers
	tr	20-10-2018	Refine FarmComponent class diagram and add description
		21-10-2018	ATAM Table for Senario 1, rework of logical view, implemntation view, change initial model, create class diagrams for farmcomponent and cloudcomponent and create component diagram for cloudcomponent
	se, rb	20-10-2018	Refined risks
	se	21-10-2018	Added section 8.2.5 Use case view + report formating of various sections
	rb	21-10-2018	7.5 Added server specifications
		21-10-2018	8.2.5 added deployment diagram
	ab	21-10-2018	8.3 software decision
		21-10-2018	7.4 Added the diagram and text in the document
0.8.0	tr	28-10-2018	Change hardware overview diagram
			Change class diagram of CloudComponent
			Rework implementation view of CloudComponent including diagram
	se	28-10-2018	Updated the outdated hardware specs (chapters 3.8, 7.2, 7.5)
			Removed FR table, added more FR under GUI HLR (ch. 4.6.1)
	ab	28-10-2018	Created process view and Sequence diagrams in chapter 8.2.4. Updated software decision
	ALL	10-11-2018	Consistency updates in chapters 1-4.5
	ALL	11-11-2018	Consistency updates in chapter 4.6-10
	rb	11-11-2018	Refactor deployment view chapter 8
	tr	11-11-2018	Refactor ATAM tables

Table 2: Changes in the document

Contents

List of Figures

List of Tables

1 Introduction

We are a software architecture team in Farmvolution which specializes in designing and developing smart systems for livestock production and management. This document seeks to provides an overview on how the architecture of the smart farming system Farmvolution.

Farmvolution enables health monitoring for livestock, livestock reporting, and smart automated farm process management in the are of livestock husbandry. The architectural description of this report helps software developer and hardware engineers during the implementation of the system. Furthermore, business stakeholder retrieve insights on the relationships to the domain field and its stakeholders, and teamleaders find useful information on the requirements of the developers in either software or hardware manners.

2 System Context

Food production in the 21st century is a crucial issue since conventional food production process will not be able to feed the growing world population without causing major problems for the following generations. Most countries have an economy that is dependent on agriculture – either in a small or big way. From employment generation to contribution to National Income, agriculture is important. In 2010, around 25 million people were regularly engaged in agricultural work in the European Union, of which 58% were working full time on farms [3]. In addition, even today's agriculture has to cope with difficulties caused by the globalization. Advancements in technologies such as Artificial intelligence and Cloud infrastructure leads to opportunities for optimizing and automating farm processes. In order to support farmers all over the world, smart farming systems are developed to reduce production costs and improve management of farms.

These technologies that help with the automation of the farm processes are rapidly being integrated into the crop sector. However, for the livestock sector, many processes are not automated yet. The automation of these processes can minimize the loss of output and optimize the usage of resources.

Clinical techniques for monitoring livestock health are insufficient, as they provide only sporadic information and require too much investment in terms of time and cost of manual labor. A sophisticated system capable of continuously assessing the health of individual animals, aggregating these data, and reporting the results to stakeholder and can improve quality and quantity of the livestock products.

Health monitoring in particular would benefit livestock farmers. For instance, the average dairy farmer loses between $20000 and $60000 worth of milk due to the use of antibiotics[4]. With early detection of disease it is possible to use natural healing methods rather than antibiotics.
The scope in this system is limited to health monitoring and livestock management for dairy cattle, beef cattle and pigs. For this system, we consider farms located in the Netherlands where there is sufficiently good access to the internet and electricity network.

3 Architectural relevant business information

This chapter describes the business planning of the product. Sections 3.1 - 3.9 provides insight to the feasibility of building this product. Initially, the Business vision and Business rationale are described. Then, the possible ways of extending the system (Section 3.4 are discussed and finally, a financial model (Section 3.8) and the current competitors (Section 3.9 - Competitors) are presented.

3.1 Business vision

In order to determine Farmvolution's strengths and weaknesses, as well as the opportunities and threats that the system might face, the following SWOT analysis has been created. This helps the architects to focus on the strengths, minimize threats, and take the greatest possible advantage of available opportunities.

The SWOT analysis of the Farmvolution is illustrated in Table 3.

Strengths - Automation of health monitoring - Flexibility w.r.t. implementation of new features - User friendly GUI - Reduction in production loss due to accurate prediction of diseases	**Weaknesses** - Some countries might have law restrictions for the usage of some sensors/techniques - The system might not be affordable to every farmer on the market
Opportunities - Farmers are looking for ways to automate their farming techniques - Technology allows for more efficient production rate	**Threats** - Strong competition in the livestock domain - Change in customer behavior can render livestock farming obsolete

Table 3: SWOT - Analysis

The Farmvolution system provides a new way to support animal farmers around the world by providing better livestock production and management. This is achieved with the implementation of the following four main features:

1. **Health monitoring:** Farmvolution detects diseases in individual animals using visual and audio data combined with advanced machine learning algorithms. The details of this feature are described in Section 3.3.1

2. **Livestock monitoring:** The system monitors the parameters of livestock such as nutrition behaviour, milk produced, etc. The details of the feature are discussed in Section 3.3.2.

3. **Reporting:** The system provides production statistics to farmers and third parties such as the food manufacturers, in order to predict the supply of goods produced by the farm, These reports are generated periodically and can be subscribed to by the users that are interested in receiving them.

4. **Notification:** The farm worker/farm owner are notified of all the relevant events such as disease detection and report generation.

The unique selling points of the Farmvolution system can be summarized as:

1. **Scalability:** The system's architecture allows the integration of new components and functions in the domain model. The architecture facilitates a scalable solution in order to serve an increasing number of farms.

2. **High performance and accuracy:** The system guarantees timely and accurate disease detection and guidance. The system uses pre-trained machine learning models [9] [2] tested on real world data sets for the detection of diseases. The models are updated periodically to guarantee high correctness, precision and reliability of the disease detection.

3. **Automatic detection of diseases:** The system is capable of automatic detection of disease by applying machine learning algorithms to video and audio sensor data. After detection, an immediate notification is sent to the farmers which includes relevant information about the affected animal.

4. **Resilience in components and resources:** High availability of the monitoring, notification and reporting activities is realized through high fault tolerance of the different components. An innovative mechanism is used to prevent the loss of data.

3.2 Business rationale

Farmvolutions mission statement is to limit the production loss caused by diseases by detecting them in an early stage. Therefore, diseases can be treated using different methods than antibiotics or other substances that are not allowed to be present in livestock products. This in turn reduces waste of food. The automation of health and livestock monitoring also has the benefit of reducing operating costs of farms.
To achieve this, the system is planned to have a pioneering role in the livestock production and management market, among which the market for livestock production in particular, by offering early and correct detection of diseases in animals.

3.3 Product/Service description

Farmvolution is a system that offers farmers a way to improve productivity and manage dairy cattle and pigs. Farmvolution provides an adaptive architecture for farm processes in order to cater to a changing needs of the livestock industry. Farmvolution's products and services are listed in the subsections below.

3.3.1 Health Monitoring:

For health management of animals to be effective the primary aim must be to prevent the spread of disease, to recognize the presence of disease at an early stage, and to treat all individual animals that are infected with parasites as soon as possible and before the disease manifests itself as a serious condition. The key principles of health management are:

1. Prevention of disease

2. Early recognition of disease

3. Early treatment of disease

For pigs and cattle, the following health monitoring methods are employed by the system:

Dairy Cattle
Automatic detection of disease called mastitis in dairy cattle using thermal imaging cameras. Mastitis is a persistent inflammation in the cow's udders. This potentially fatal mammary gland infection is the most common disease in dairy cattle. It can be treated easily if diagnosed in an early stage. Advanced algorithms are used to automatically analyze data provided by the thermal cameras to detect the
disease [9].

Pigs
Most disease in pigs are respiratory. With some diseases you can detect signs just three hours after infection. The system will monitoring audio data to detect coughs in pigs and distinguish those cause by disease from regular coughing sounds [2]. If such a sick cough is detected then the system will notify the farmer through the dashboard, who can go out and decide if they need to call a veterinarian.

3.3.2 Livestock Monitoring

For pigs and cattle, the following livestock monitoring methods are employed by the system:

Dairy Cattle
For Dairy cattle features such as the amount of milk produced are monitored and aggregated with the reporting system. The farmer access this information using a web-interface.

Pigs
Livestock requirement monitoring for group-housed pigs using Depth-Enabled Multi-Object Tracking [8]. The combination of movement data and physical parameters make it possible for the system to monitor growth rate, classify aggression, and detect early signs of compromised health allowing for individualized care and management in large group settings.

3.4 System Evolution

Upon becoming broadly accepted in the Netherlands, the system will go through a period of both expansion and extension. One of the goals of the company is to penetrate international markets. The expansion in different countries and continents will be achieved without changes in the core components. Example of an expected change is the addition of different language packages required for the respective countries.

Furthermore updates will be made to the web interface, the number of features will be increased based on the requirements provided by the stakeholders.The farmvolution system plans on extending the system into crop farming.

3.5 Target audience

The target customer of the Farmvolution system has a high acceptance of new methodologies and technologies for better management of livestock. The target customers are the dairy cattle farmers, pig farmers and food distributors associated with these farms in the Netherlands. The Farmvolution system is suitable for all types of milking parlours (robot, carousal, herringbone) and group-housed pig farms. The target customer is aware about the possibilities of early detection of diseases, reduction of antibiotics costs and improved productivity for animals.

3.6 Business and Domain model

This section provides an overview on the business model as well as the domain model of the Farmvolution System.

3.6.1 Business Model

Figure 1 shows the business model containing the main entities which are relevant to the system.

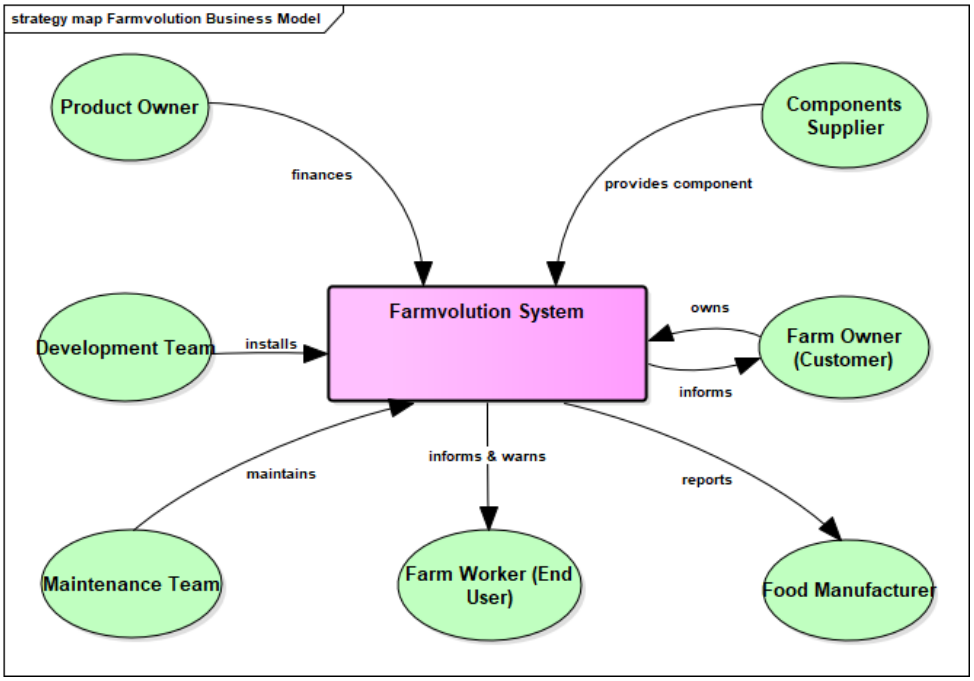

Figure 1: Business Model

1. **Product Owner** - The product owner is the investor and initiator of the product
2. **Components Supplier** - Suppliers include all the parties which deliver the components and sensors that build up the system.
3. **Farm Owner (Customer)** - The customers are the buyers of the system - anticipated and targeted to be farm owners.
4. **Food Manufacturer** - Food Manufacturers are the stakeholders which receive livestock products from farmers for manufacturing of their product.
5. **Farm Worker (End User)** - The farm workers are those stakeholders that do not necessarily pay for the system, but actually utilize the system on daily basis.
6. **Maintenance Team** - The maintenance team needs to maintain the product over its lifetime and to ensure that the product remains maintainable
7. **Development Team** - The development team is concerned with developing and installing the product

3.6.2 Domain model

Figure 2 shows an overview of Farmvolution's domain model. The model is divided into structural and behavioural elements. The relationship between them are as follows:

1. **MonitoringService** - is a behavior element of the domain model which monitors MonitoringSubjects. An Animal and ExternalData have the role of a MonitoringSubject. Animal is an abstraction for Pig and Cow and defines certain attributes for the specific type of animal. ExternalData represents data from farm devices that already exists at the farm or will be integrated. MonitoringService creates a MonitoringResult.

2. **ReportingService** - is a behaviour element used for generating Reports based on MonitoringResults.

3. **AutomatedProcessService** - is a behaviour element used to create AutomateProcesses. This processes are defined by a Farmer.

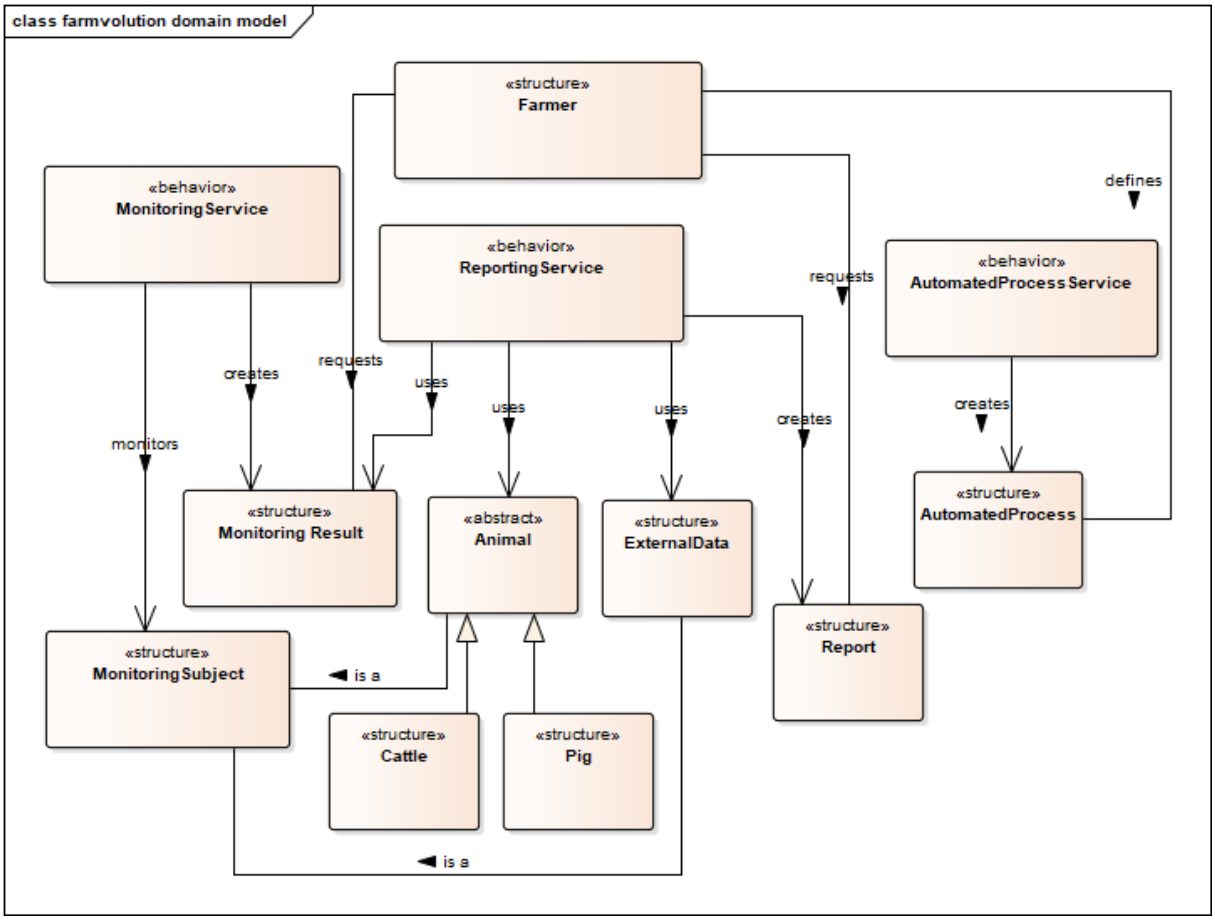

Figure 2: Domain Model

3.7 Roadmap

Farmvolution is expected to launch the minimum viable product in the first quarter of 2020. It is estimated that the development time of a mature system, with all the components, sensors, interfaces, protocols, testing, training of machine learning models and the software will take approximately one year. The country of initial deployment is going to be the Netherlands, because dairy cattle and pig

farming is carried out on a large scale. Furthermore, the availability of a high speed internet connection is one of the best in Europe due to the high population density and flat, soft soil. These conditions make the Netherlands an ideal starting location for Farmvolution.

Once the product has been tested and refined in the Netherlands in the year 2020, the system will be adapted for launch in other European countries. This expansion will begin in Q1 2012.

The roadmap anticipates to start delivering the product first to milking parlors and Group-Housed Pigs. After a successful market launch of these systems acros Europe, it is planned to extend the target domain to include the crop sector. Farmvolution's crop monitoring solution is planned to be delivered in Q1 2022 in Netherlands and in Q1 2023 in Europe.

Period	**Location**	**Milestone**
2020 Q1	Netherlands	Launch Farmvolution system for dairy cattle and pigs
2021 Q1	Europe	Launch Farmvolution system for dairy cattle and pigs
2022 Q1	Netherlands	Launch Farmvolution system for crops
2023 Q1	Europe	Launch Farmvolution system for crops

Table 4: Roadmap Overview

3.8 Financial model

This section describes the financial model of the system. The total costs of the system amount to €55,000 for the software architecture over a period of 10 weeks. This is then followed by weekly costs of €15,200 for the development, €7,600 for the maintainers and €45.36 for services.

The hardware costs are based on the initial package, as well as 2 monitoring packaging. The base package is €752.67, which contains the default local server required for any subsequent packages. Depending on the type of farm, the users can also opt for the cow monitoring package, which is €644,82 or the pig monitoring package, which is €20,361.49.

Together, the hardware costs for a dairy farm amount to €1397,49, the hardware costs for a group housed pig farm amount to €21,114.16 and the hardware costs for both pig and cow farms amount to €21758,98.

Subsections 3.8.1-3.8.5 provide detailed inspection of the costs incurred.

3.8.1 Software architecture costs

The main architecture is projected for a period of 10 weeks. The team involved in the architecture consists of 5 members. Each member spends an average of 10 hours per week, with an hourly wage of €110. Thus, the total cost of the software architecture is:

$$5(architects) \cdot 10(weeks) \cdot 10(h/week) \cdot 110(€/hour) = €55{,}000$$

3.8.2 Development costs

It is estimated that a team of 10 software developers is needed to efficiently build the system in the least amount of time. Each software developer works an average of 38 hours per week, with an average salary of €40 per hour. Thus, the weekly cost for the software development is:

$$10(developers) \cdot 38(h/week) \cdot 40(€/hour) = €15{,}200$$

3.8.3 Maintenance costs

After the deployment of the system, maintenance must be carried out in order to upgrade system's modules, change outdated sensors with the latest ones and update the software based on user's feedback.

This can be achieved by having a team of 10 maintainers that can move on site whenever is needed. Similar to the developers, each maintainer will work approximately €38 hour per week, with an hourly wage of €20. Thus, the weekly cost for the maintenance is:

$$10(maintainers) \cdot 38(h/week) \cdot 20(€/hour) = €7{,}600$$

3.8.4 Services costs

The system makes use of 3rd parties for cloud database (DB) and processing of the data collected using machine learning algorithms. These are achieved using Amazon Web Services (AWS). For the purpose of the system, it is enough to acquire a General Purpose m5.xlarge package[1], which amounts to €0.27 per hour. This is equivalent to €45.36 per week or €2358.72 per year.

3.8.5 Hardware costs

The hardware modules required to build the system are already available on the market, so there is no need to develop customized equipment. The quantity of the sensors needed for the proper usage of the system depends upon the dimensions of the farm. The larger the farm, the more sensors are required to monitor the entire livestock population. At the very least, each farm requires a local server with 500GB of SSD, 16GB of DDR4 RAM and an Intel Xeon e3 v6 processor (4 cores, 8 threads).

In addition to this basic server, some additional hardware is required which varies based on the target animal. The following components have been determined for the average size of cattle and pig farms in the Netherlands. For dairy cattle, the average amount of animals is 200 cows while for pigs the average number of group housed pigs is 2500.

For each entrance to a milking parlour 2 Forward Looking Infrared Radiometer (FLIR) thermal cameras and 1 bridge device are needed to capture and pre-process the collected data). In addition to this, around

[1] `https://aws.amazon.com/emr/pricing/`

100MB of local storage is required to store the thermal image data that is collected by the FLIR cameras.

The average number of pigs per housing setup is 15 and requires 1 kinect V2 camera and 1 Raspberry Pi. Therefore, the average farm requires 150 kinect v2 camera's and 150 RaspberryPi bridges. To connect all of these devices to the local server, an estimated 15.000 meters of 1Gb/s Ethernet cable is required. To store and process all this data, the local server requires an upgrade of 16 GB of extra RAM (for 32GB in total), as well as 5 10TB hard drives that were specifically designed for video monitoring. Section 7.5 shows a detailed calculation on how these hardware needs were determined.

The FLIR TG165 thermal imaging system[2] is used to measure the animals' body temperature. For cows, measuring the temperature of the udder is used to detect diseases (such as Masitis) up to 4 days in advance[3]. The cost for this camera is €284.91.

The bridge that connects the FLIR cameras can be a simple Raspberry PI 3B device[4]. The same bridge can also be used to connect other devices already present in the farm, such as a milking sensor.

Kinect v2 cameras[5] are used to measure the animals intake of food and water. For group-housed animals the cameras capture depth images, which are stored in the database monitor and analyze eating habits of the livestock. The cost of 1 camera is €84.99, so the cost of 2 cameras amounts to €169.98.

Tables 5 - 7 provide an overview of the aforementioned hardware and costs.

Sensors	**Quantity**	**Cost per Unit**	**Total Cost**
Samsung 860 EVO 500GB 2,5 inch[6]	1	€89.99	€89.99
Corsair Vengeance LPX 16 GB DDR4[7]	1	€167.99	€167.99
SUPERMICRO MBD-X11SSL-F-O Micro ATX Server Motherboard[8]	1	€189.99	€189.99
Intel Xeon E3-1245 v6[9]	1	€304.70	€304.70
Total Cost			**€752.67**

Table 5: Financial Model for the default local server

[2] https://www.amazon.de/FLIR-W%C3%A4rmebild-IR-Pyrometer-1-St%C3%BCck-TG165/dp/B00NXJDQV0/ref=sr_1_3

[3] https://www.flir.eu/discover/instruments/condition-monitoring/automatic-health-check-in-dairy-farms-\
using-flir-thermal-imaging-cameras/

[4] urlhttps://www.alibaba.com/product-detail/Hotsale-Raspberry-Pi-3-Model-B$_6$0815583794.*html*

[5] https://www.ebay.com/itm/Microsoft-Kinect-Sensor-for-Windows-1517-Commercial-Developers-Unit-in-Box-/\
312230432320

Sensors	Quantity	Cost per Unit	Total Cost
FLIR Thermal Camera[10]	2	€284.91	€589,82
Raspberry Pi Bridge[11]	1	€35.00	€35.00
1Gb/s Ethernet Cable (per meter)[12]	500	€0.04	€20.00
Total Cost			**€644.82**

Table 6: Financial Model for cow monitoring

Sensors	Quantity	Cost per Unit	Total Cost
Seagate SkyHawk 10TB 3,5 inch[13]	5	€319.00	€1595.00
Corsair Vengeance LPX 16 GB DDR4[14]	1	€167.99	€167.99
Kinect v2 Camera[15]	150	€84.99	€12748.5
Raspberry Pi Bridge footnotehttps://www.alibaba.com/product-detail/Hotsale-Raspberry-Pi-3-Model-B_60815583794.html	150	€35.00	€5250.00
1Gb/s Ethernet Cable (per meter)[16]	15000	€0.04	€600.00
Total Cost			**€20361.49**

Table 7: Financial Model for pig monitoring

3.9 Competitors

The market for agricultural-tech is growing with development of novel technologies. There are some large scale and small scale competitors that also offer a system for livestock production and management that are currently in development. The most prominent of these competitors:

1. **Bovcontrol**[17]
   - The company focuses on provides a livestock manager to cattle farmers to keep track of the herds using cloud technology. Bovcontrol tracks inventory, vaccinations, nutrition needs and more.
2. **DeLaval**[18]
   - DeLaval is a leading producer of dairy and farming machinery and is specialized in automating milking. Herd management, providing cooling, storage and testing solutions.
3. **Nedap**[19]
   - Nedap Livestock Management is the global leader in farming automation using individual animal identification. Their solutions allow animals to live in groups, yet still receive the individual attention they need. Individual care supports optimal condition for excellent productivity.

[17] https://www.bovcontrol.com/
[18] https://www.delaval.com/nl/
[19] https://www.nedap-livestockmanagement.com/about-us/nedap-livestock-management/

4 Requirements

This chapter describes the various stakeholders of Farmvolution and list their concerns. From there a number of use cases and key drivers are derived. This is then followed by a list of functional and non functional requirements. The final subsection is concerned with risk assessment.

4.1 Architectural vision

The Farmvolution system operates in three main components: the farm component, the cloud component and the web GUI component. Below is a very brief summary of the architectural vision that illustrates how these web, cloud and farm components will interact. The visualization of this vision has been depicted in Figure 3.

The sensors are the primary source of data for Farmvolution. These are located in stables as depicted in Figure 3. Three types of farm components are illustrated:

1. A sensor array for pigs with video, audio and depth sensing capabilities.
2. A sensor array for cows with thermal cameras.
3. A local system that serves as endpoint for communication with the cloud services.

Data collected by these sensors is pre-processed at the farm and is processed and stored in the cloud, where services are easily scalable to suit different sizes of farms. Processed data is made available through an API, so that UI applications and food manufacturers can access them.

The UI component consists of an intuitive web based interface for farmers that operates on their personal devices. This is where the end users define monitoring rules and receive reports and notifications about the data that is collected.

Sensors are present on the farm in two different forms: one for cows and one for pigs. Kinect v2 cameras are present at pig pens, where they keep track of the location, food consumption, water consumption. The cameras are also able to perform weight estimates based on the observed dimensions. This monitoring is continuous.

Cows are monitored periodically, specifically when they enter the milking area. A milking area could have multiple entrances and a varying setup. Therefore, a single setup is supplied for each entrance of the milking area. This setup consists of 2 thermal cameras, an Radio-Frequency IDentification (RFID).

Data collected by these sensors goes through a local machine that virtualizes collected data. From there it is processed and monitored in the cloud. When information is provided to the farmer (i.e. when a rule is violated, or a report is requested), then the cloud service pushes a notification the web interface where to farmer can view it.

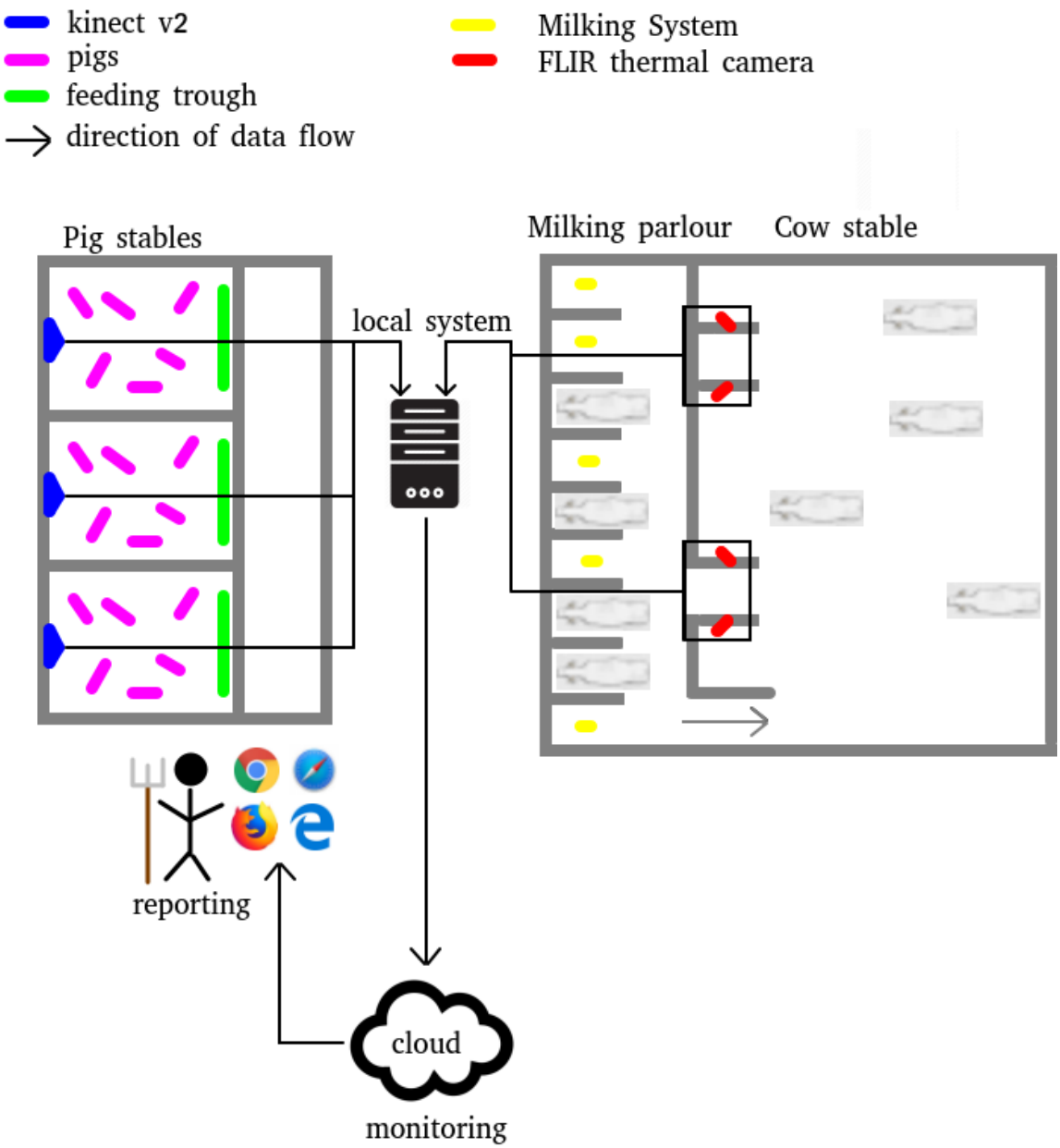

Figure 3: Architectural Vision

4.2 Stakeholders

This section describes the main stakeholders that have relevant interests in the Farmvolution system. Each stakeholder has a list of relevant Quality Attributes (QA). Table 8 displays the concern matrix, in which every QA is measured in terms of the weighted stakeholder's interest. More information regarding the concern matrix can be found in section 4.2.1 (Concern Matrix).

The main stakeholders of the Farmvolution system are:

1. **Farm owners** - The farm owners will purchase the Farmvolution system and will compensate the company for periodic maintaining of its components.
   (a) **Profitability** - Farm owners want the system to help them improve profits of the farm.
   (b) **Affordability** - The product should not exceed farm owner's budget.
   (c) **Usability** - The system should have high usability in order for the farmer to interpret the information appropriately.
   (d) **Availability** - The system should be highly available to the farmer at any time.
   (e) **Reliability** - The system should be fault tolerant, that so the farmer don't have to spend much time on fixing the issues of the system.
   (f) **Performance** - System should analyze data accurately, since farmer will use the data to make critical decisions.
2. **Product Owner** - The product owner (PO) provides finance and oversees the development of the product.
   (a) **Profitability** - This is the main concern of the Product Owner, because he is the one purchasing the system so the PO wants to make sure that it will bring him value.
   (b) **Affordability** - This is also a concern of the PO, since an affordable system has a larger number of potential customers.
   (c) **Adaptability** - This is relevant because it reduces the cost of adding new components and features in the future. The PO will not have to buy new system since he can add new features.
   (d) **Performance** - This is important for customer satisfaction, therefore the PO will be sure to have satisfied costumers and he won't have to deal with complains.
3. **Farm workers** - The farm workers will use the Farmvolution system to receive information and notifications when a disease has been detected in an animal to take required measures.
   (a) **Usability** - This is the primary concern of the farm workers, because he will be the one using the system in the first place
   (b) **Availability** - This is relevant because an unavailable system can hinder the work that a farm worker has to do.
   (c) **Reliability** - This is important for the farm worker because an incorrect measurement can lead to a waste of time or product.
   (d) **Performance** - This because the farm worker will be using the system on daily basis the system should have high performance, ease of use, the ability to communicate with other systems for updates and that the system should adapt to changing user needs

4. **Food manufacturers** - They are third party companies such as milk, cheese and bacon industries which manufacture food by purchasing products from farmers. They would like to be able to retrieve information about the logistic demands of the farm. Think of factors such as the amount of animals that should be transported, or the quantity of milk that is ready to be shipped.

   (a) **Availability** - This is in the food manufacturers' interest because they cannot rely on the system if data is unavailable.

   (b) **Performance** - because with accurate data they can manage their inventory, logistics and predict their production accordingly.

   (c) **Integrability** - Food manufacturers want to easily integrate data from Farmvolution into their own system.

5. **Development team** - The developers and installers of the software and hardware that is needed for the Farmvolution system to function correctly.

   (a) **Adaptability, Modularity** - development teams want to easily add new features and technologies.

   (b) **Testability** - developers want the system to operate as expected.

6. **Maintenance team** - The Maintainers are responsible for the maintenance of the software, hardware components and detection of errors or malfunction that are concerned with the system, as well as the back-end, front-end and customer support of all peripheral systems such as web-interface.

   (a) **Modifiability** - Reducing the amount of work needed to adapt the system to new customer needs, the maintenance team will spend less time fixing the issues.

   (b) **Performance, Availability, Reliability** - A reliable system with good performance and little downtime requires less maintenance, therefore less problems to deal with for the maintenance team.

   (c) **Modularity** - This makes it easy to replace broken sensors, so the maintenance team will not have to do major changes during an upgrade.

4.2.1 Concern Matrix

This subsection describes the value of the stakeholders' interests in form a concern matrix. Thus, the relevance of a QA is measured for each stakeholder and ranked from 0 (not interested at all) to 10 (very important).

Every stakeholder has a certain weight, measured as a percentage, based on their importance for the Farmvolution's architecture. The stakeholders mentioned in section 4.2 (Stakeholders) have the following weight:

1. Farm Owner (FO) has a weight of 100%;
2. Product Owner (PO) has a weight of 100%;
3. Farm Worker (FW) has a weight of 90%;
4. Food Manufacturer (FM) has a weight of 75%;
5. Maintenance team (MT) has a weight of 65%;
6. Development team (DT) has a weight of 60%;

By summing the weighted relevance values of all stakeholders for each QA, a score is obtained, which determines the most important quality attributes of the system. The 3 most important QA determine the main key drivers of the Farmvolution system. In order to determine the key drivers, the following QA's were considered:

1. Timeliness - The ability to satisfy requirements within a given time-frame;
2. Performance - Measure for the efficiency of a system;
3. Reliability - Ability of a computer-related hardware or software component to consistently perform according to its specifications;
4. Availability - Measured by the percentage of time where Farmvolution is able to satisfy requirements, relative to its expected time;
5. Usability - Level of easiness required to learn how to use the system;
6. Modifiability - The ability to future technologies and changes in requirements;
7. Safety - Indicated by the chance where harm or injury could occur to those that use the system;
8. Security - The ability of the system to guarantee confidentiality, integrity and availability of user-data;
9. Portability - The easiness to adapt the system to new environments such as different types of farm;
10. Reusability - Indication of the portion of written code that can easily be reused in order to meet new requirements;
11. Integrability - The ability to integrate Farmvolution with other systems through the use of protocols and signals;
12. Interoperability - The ability to communicate with other systems due to similarity in semantic context;
13. Testability - The ability to verify whether Farmvolution meets its requirements through the use of techniques such as unit tests.

Note: other possible quality attributes could be profitability, time to market (TTM), cost, return of investment (ROI), total cost of ownership (TCO), lifetime, integration or marketing. These quality attributes are not included in the concern matrix because they are not architecturally significant for the Farmvolution system.

Table 8 shows an extract of the aforementioned QA's. The full table can be found in Appendix Full Concern Matrix A.1 (Table 49).

	Weight (%)	**Timeliness**	**Performance**	**Reliability**	**Availability**	**Usability**	**Modifiability**	**Safety**	**Security**
FO	100.00	5	10	10	10	8	4	7	7
PO	100.00	7	10	9	9	8	7	4	7
FW	90.00	8	7	7	9	10	5	8	8
FM	75.00	10	9	10	9	8	2	7	5
MT	65.00	7	7	8	9	5	9	4	7
DT	60.00	3	8	8	6	7	7	4	7
Total		34.85	**42.4**	**42.2**	**43.3**	38.45	27.05	32.45	33.7

Table 8: Concern Matrix

4.3 Key Drivers

The concern matrix displayed in Table 8 (section 4.2.1) shows that main key drivers are Availability, Performance and Reliability.

Availability

The availability of a system is the ratio of time that the system is working as intended over the time that it should be operating as intended. It is important for a system to be available for constant monitoring and detection of disease in a given time period to the farm workers.

Performance

The main purpose of the system is to facilitate animal farming in a smart way. To achieve this, the system should be able to provide accurate, real-time information about the status of the animals (e.g. their health status). At the same time, the system should act promptly on the actions triggered by the automated process service. The accuracy and timeliness of the notifications are tremendously important factors for the success of Farmvolution.

Reliability

Reliability is very important for the Farmvolution, since wrong decisions/suggestions/calculations are made when the system is not working properly. If the system is unreliable, the customer may lose confidence in the product and probably buy a competitor's product. Additionally, selling an unreliable system may cost Farmvolution a lot of money for maintenance or support.

4.4 High-level Requirements (HLR)

The following requirements represent the system's high-level functionality. Each high-level requirement will be refined in detail by various specific and technical requirements.

Label	Importance	Description
Monitoring health state		
HLR1	**Must**	The system is able to monitor the health state of animals. Since there are many different ways to monitor health state for different kinds of animals, Farmvolution system is able to implement several techniques to monitor the health state. After the system has check the health state of a particular animal and or a group of animals, it should be able to send a report to the farmer. If the system detects a disease it should be able to send a notification immediately
Livestock reporting		
HLR2	**Must**	The system is able to measure livestock information such as weigh of animals, feeding and drinking behavior, or collected milk, over a certain time period and send a report to the farmer. The farmer should be able to customize these reports.
Provide farm information		
HLR3	**Must**	The system is able to measure farm production information such as monthly amount of milk production or weight of animals. This information can then be retrieved by the food manufacturers. An Application Programming Interface (API) provides an entry point to the system.
Define automated farm processes		
HLR4	**Must**	The farmer is able to add farm processes depending on data of several sensors and trigger actuators if necessary. One example could be to weight an animal once a day and increase the amount of food if the animals weight is to low. The farmer should be able to add/adjust the process through the Graphical User Interface (GUI) dashboard of the Farmvolution system.
GUI		
HLR5	**Must**	Farmvolution system provides a GUI on which Farm workers and Farm owners can use the main functionality. The GUI enables access to the system.

Table 9: High-level requirements of Farmvolution

4.5 Stories and Use-cases

Figure 4 indicates the general Use Cases of Farmvolution.

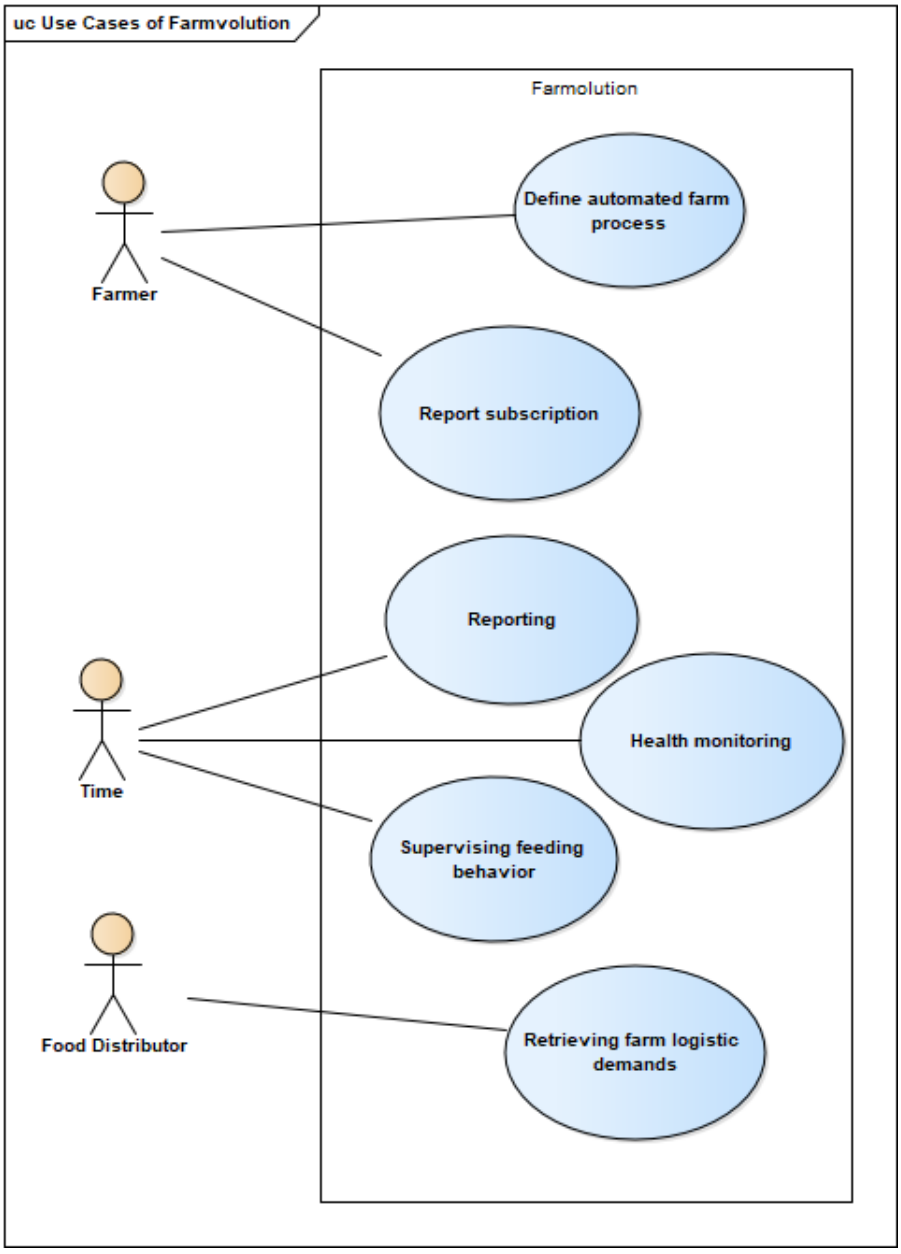

Figure 4: Use Cases

4.5.1 Use Case 01: Farmer defines automated farm process

Primary actor: Farmer (owner, worker)
Preconditions: All animals can be scanned by the correct sensor that delivers adequate data to monitor farm process
Scope: General usage of system
Level: counts for all animals that fulfill a certain condition

Main Success Scenario:

1. Farmer accesses Farm Process Area of Dashboard and clicks add new rule
2. Farmer set demanded settings
3. System evaluates if process is applicable (all required sensors, data, devices are available)
4. System adds process to existing processes and displays success message on Dashboard

Extensions

1.a A new device or sensor is required

2.a System needs access to third party system which requires particular authorization

2.b Evaluation of process fails

- 2.b.1 System detects a problem
- 2.b.2 System displays error message
- 2.b.3 If possible: System provides a solution
- 2.b.4 Farmer can check setting to ensure demanded settings were add correctly

Sub-variations: Downloading process

1. Farmer accesses shared farm process on certain online platform
2. System checks if all required hardware and information is available
3. System downloads farm process from online platform

4.5.2 Use Case 02: Health Monitoring

Primary actor: Time
Preconditions: Since this Use Case is a general one and can be implemented for certain Health monitoring procedures we assume all required sensors to be installed
Scope: Health monitoring

Main Success Scenario:

1. Farmvolution detects subject to monitor
2. Farmvolution monitors subject
3. Farmvolution stores result of monitoring process

Extensions:

2.a Subject fulfills a certain condition

2.a.1 Farmvolution detects condition

2.a.2 Farmvolution notifies Farmer

4.5.3 Use Case 03: Supervising drinking and feeding behavior

Primary actor: Time
Preconditions: Pigs are registered in the system with an RFID chip.
Scope: Supervising behavior
Level: Pig package, user level

Main Success Scenario:

1. Senor collects data for drinking and feeding behaviour
2. FarmComponent sends data to CloudComponent
3. CloudComponent analyzes behaviour
4. Result is stored

Extensions:

1.a System is not responding as expected

3.a System cannot detect drinking or feeding behavior

4.5.4 Use Case 04: Reporting

Primary actor: Time
Preconditions: All required sensors are installed
Level: reporting level

Main Success Scenario:

1. System collects certain information for one animal
2. System processes the received data and creates a single report for that animal regarding the particular field of information (e.g. amount of milk of an cow, e.g. weight of a pig)
3. System stores report

Extensions:

2.a Processed data has to be aggregated in order to create a new report regarding aggregated information for a group of animals (e.g. amount of milk of one milking session)

4.5.5 Use Case 05: Report subscription

Primary actor: Farmer
Preconditions: Reports are generated from Reporting Use Case and be available
Scope: General usage of system
Level: user level

Main Success Scenario:

1. Farmer accesses Reporting Area of Dashboard and clicks subscribe to Report
2. Farmer selects report
3. Farmer configures rules when and which information of report should be send

Extensions:

1.a System is not responding as expected

2.a Demanded report is not available

2.a.1 Trigger demanded report creation

4.5.6 Use Case 06: Retrieve logistic demands

Primary actor: Food Manufacturer
Preconditions: required information are available as a report
Level: demanding logistics level

Main Success Scenario:

1. Food Manufacturers calls API of Farmvolution and requests information (e.g. amount of milk of last week)
2. Farmvolution sends certain report or some information of it

Extensions:

2.a Demanded information are not considered in stored reports and have to be generated from reports (e.g. expected amount of meat of next slaughter (system only knows weight of animals, but might be able to generate expected amount of meat))

4.6 Functional requirements

This section presents the functional requirements of the Farmvolution. The functional requirements are given a label which will be used to refer to them later. A distinction is made between must-have and optional requirements. This is indicated in the second column. The third column contains a short description of the requirement, as well as which actor is associated with that requirement.

4.6.1 Mapping FR's into the HLR's

Tables 10 - 14 provide the mapping of the FR's into the HLR's. The numbering schema of the FR's is as follows: the first number is related to the HLR, while the second number is the index of the FR.

Label	Importance	Description
FR1-1	**Must**	The system must be able to measure livestock's drinking and feeding behaviour.
FR1-2	**Must**	The system must be able to use audio recording to detect when pigs are coughing.
FR1-3	**Must**	The system must be able to periodically take thermal images of the udder of each cow.
FR1-4	**Must**	It must be possible to weigh animals.
FR1-5	**Must**	The system must be able to analyze thermal images using cloud services to detect diseases in cattle.
FR1-6	**Must**	The system must be able to analyze audio data using cloud services to detect diseases in pigs.

Table 10: Monitoring health state HLR1

Label	Importance	Description
FR2-1	**Must**	The system must be able to receive milking data collected by the farms.
FR2-2	**Must**	The farmer must be able to subscribe to periodic reports that are generated by the system.
FR2-3	**Must**	The system must notify the farmer when a disease is detected in pigs and cattle.

Table 11: Livestock reporting HLR2

Label	Importance	Description
FR3-1	**Must**	Food Manufacturers must be able to receive product information from Farmvolution.
FR3-2	**Must**	A farm owner must be able to authorize a third party system to use data recorded on the farm. This is done by entering a domain from which data may be requested or submitted.

Table 12: Provide farm information HLR3

Label	Importance	Description
FR4-1	**Must**	The farm owner and worker must be able to define automated farm processes for cow monitoring.
FR4-2	**Must**	The farm owner must be able to define automated farm processes for the information reported to third parties.

Table 13: Define automated farm process HLR4

Label	Importance	Description
FR5-1	Must	The GUI must have a dashboard to be used by the farm owners and farm workers.
FR5-2	Must	The GUI must have a reporting area in the dashboard.
FR5-3	Must	The GUI must have an automated farm process management in the dashboard.
FR5-4	Must	New notifications should be displayed automatically in the GUI.
FR5-5	Optional	The homepage of the GUI should be customizable by choosing which modules are displayed on it.

Table 14: GUI related HLR5

4.7 Non-functional requirements (NFR)

This section describes the mapping of the non-functional requirements to the aforementioned key drivers in section 4.3 (Key Drivers). The relevant non-functional requirements for the system are:

1. Commercial non-functional requirements
2. Technical non-functional requirements

The following two subsections (sections 4.7.1 and 4.7.2) describe these non-functional requirements in more details. Then, subsection 4.7.3 provides the mapping of the non-functional requirements to the key drivers.

4.7.1 Commercial Non-Functional Requirements (CNFR)

CNFR's are those non-functinal requirements that are related to business requirements. Table 15 shows the commercial non-functional requirements of the system. The first number is related to the HLR, while the second one is the current index of the CNFR. If an NFR does not map to a HLR then the first number is a 0.

Label	Importance	Description
NFR5-1	**Must**	The system web interface and user manual must be available in local languages and English.
NFR0-1	**Optional**	Maintenance for the system should be available on weekdays as well as on weekends
NFR0-2	**Must**	The hard drives must have a minimum lifespan of 3 years. During these three years, data recovery (if possible) in case of a failure is free of charge.

Table 15: Commercial NFR's

4.7.2 Technical Non-Functional Requirements (TNFR)

TNFR's are non-functonal requirements that specifiy criteria that can be used to judge the operation of a system under certain conditions.

4.7.3 Mapping of TNFR's to the Key Drivers

This subsection describes the mapping of the TNFR's to the aforementioned key drivers in subsection 4.3 (Key Drivers). For this, the following tables are created:

1. Table 16 - displays the NFR's related to the Availability QA.
2. Table 17 - displays the NFR's related to the Performance QA.
3. Table 18 - displays the NFR's related to the Reliability QA.

Label	Importance	Description
NFR0-3	**Must**	Failure of 1 component of the system should not lead to failure of other components or sensors. Waterfall dependencies of compontents should be avoided where possible. System-critical components that may not fail under any circumstance should have some form of redundancy to minimize the risk of failure.
NFR0-4	**Must**	Availability of the system should be high. Minimizing the duration of downtime has priority over reducing the MTTF because failure to detect anomalies for longer periods of time is riskier than the failure to detect anomalies slowly.
NFR0-5	**Must**	There should always be one database instance available.

Table 16: Availabilty NFR's

Label	Importance	Description
NFR4-1	**Must**	The system must detect rule violations within 1 minute.
NFR2-1	**Must**	Within 5 seconds of detecting a rule violation, the system must send a notification to the farmer.
NFR1-1	**Must**	Video and audio processing components should have high throughput in order to handle large volumes of streamed data.

Table 17: Performance NFR's

Label	Importance	Description
NFR0-6	**Must**	Storage for collected data should be scalable because of high resolution images that are to be stored there. Storage needs are expected to rise rapidly when Farmvolution's customer base grows.
NFR1-2	**Must**	Data should be stored locally on the farm for 1 day for redundancy purposes.
NFR1-3	**Must**	There should not be any loss of data when a component fails.

Table 18: Reliability NFR's

4.8 Evolution Requirements

This section describes the requirements applicable to the evolution of the system given in Section 10 after its initial release.

Label	Importance	Description
ER1	**Must**	The architecture of the system should allow the extension to crop packages.
ER2	**Must**	The system should be easily accessible for all country where Farmvolution has customers.
ER3	**Must**	The systems cloud component should be accessible from a mobile application

Table 19: Evolution requirements

4.9 Risk assessment

Table 50-54 (In Appendix A.2 Full Risks Tables) provides an overview of the risks that must be considered for the proper functioning of the system. Those risks don't have to do with hardware or system failure, since it is already known that it is going to happen sooner or later. The risks have to do with possible unpredictable scenarios.

5 Analysis

This section discusses the assumption made of the famrvolution system. Subsequently, the technology roadmaps are outlined and the sections is concluded with the the high level design decisions and design alternatives.

5.1 Assumptions

Following are the assumptions about the farmvolution system:

1. The farm has an internet connection and an internet provider.
2. The internet connection is reliable enough to upload and download large databases records. In case of low internet connection, the synchronization will not work properly.
3. The farm owner and the farm worker have a web-browser to access internet pages.
4. The farm has an adequate power supply for all electrical farm devices (e.g. milking machine, etc.).
5. The 3rd party systems can communicate via standard HTTP/HTTPS protocol.
6. Farm establishment is located in proper location where the signal communication between the components will not be disturbed by any other source of wireless signals.
7. There is sufficient space in the farm to install the system.
8. There is a fast internet connection to upload, the next day, gathered data over 24hour internet break down

5.2 Technology roadmap

The technology roadmap for the Farmvolution system is depicted from the Table 20.

The first iteration will consist of minimum viable product for cattle and pigs. The sensors will be used for collection of images and audio for cattle and pigs respectively and establish a communication between the sensors and local server. The collected data will be used for testing the pre-trained Machine learning models. Finally the Web UI will be defined for cattle and pig products and testing of the software and infrastructure of the farmvolution system will be carried out. By the end of 2019 Q4 this iteration should be finalized and the product can be launched in Netherlands in 2020.

The second iteration will focus on refining the existing features based on the feedback from the customers. A mobile application will be created for farm workers, farm owners and food manufacturers to view the information. The Web UI will be refined for cattle and pig products for easy usability for the customers.

Quarter	1st iteration (2019)
Q1	Minimum viable product for cattle and pigs
Q2	Test pre-trained ML models for cattle and pigs
Q3-4	Define Web UI for cattle and pig products
	2nd iteration (2020)
Q1	Refine existing features for cattle and pig products
Q2	Mobile application for cattle and pig products
Q3-4	Refine Web UI for cattle and pig products
	3rd iteration (2021)
Q1	Minimum viable product for crops
Q2	Test pre-trained ML models for crops
Q3-4	Define Web UI for crop products
	4th iteration (2022)
Q1	Refine existing features for crops
Q2	Mobile application for crop products
Q3-4	Refine Web UI for crop products

Table 20: Technology roadmap

The third iteration will consist of minimum viable product for crops. The sensors will be used for collection of images and establish a communication between the sensors, actuators and local server. The collected data will be used for testing the pre-trained Machine learning models. The Web UI will be defined for crop products and testing of the software and infrastructure will be carried out. By the end of 2020 Q4 this iteration should be finalized and the product can be launched in Netherlands in 2021.

The fourth iteration will focus on refining the existing features for crop products based on the feedback from the customers. A mobile application will be created for farm workers, farm owners and food manufacturers to view the information related to crop products. The Web UI will be refined for crop products for easy usability for the customers.

5.3 High level design decisions

Tables 22 to 28 contain the architecturally relevant high level design decisions that we made based on the stakeholder and requirements analysis in chapter 4. The goal of these decisions is to minimize the chance that one or more of the key drivers (availability, performance and reliability) cannot be satisfied due to unforeseen circumstances. Table 21 shows an overview of the design decisions and the key drivers that they contribute to.

Decision #	**Availability**	**Performance**	**Reliability**
Decision 01	yes	no	yes
Decision 02	yes	no	yes
Decision 03	yes	yes	no
Decision 04	no	no	yes
Decision 05	no	no	yes
Decision 06	no	yes	no
Decision 07	yes	no	no

Table 21: Mapping of decisions to key drivers.

Name	**Solution01: Local backup of critical cloud components**
Decision	#1
Problem/Issue	If the farm loses access to the cloud services, the critical components that could directly hinder normal farm operation should be executed locally
Decision	We decided to use a small local server that is able to do basic rule evaluation on a replicated dataset that is synchronized once every hour.
Alternatives	An alternative solution would be to move all processing and storage over to a local server.
Arguments	The alternative solution has the downside that a larger up-front payment must be made by the farmer, in order to justify the purchase of a sufficient local server. Furthermore, this would imply an increase in electricity cost for the farmer. A cloud solution for heavy tasks is more easily scaleable.

Table 22: Decision 1: solution for loss of internet access

Name	**Solution02: Redundancy to avoid failure of critical component**
Decision	#2
Status	Approved
Problem/Issue	Critical components might fail at run-time
Decision	Redundant storage with appropriate process monitoring and synchronization protects against system failure when the storage is suddenly not available. In this case the StorageMonitoring component switches the connection to the backup storage. The RuleObserver, which detects changes of data, does not recognize it since a StorageWrapper hides this process.
Alternatives	If the storage is not available at a particular time, the system may attempt to access it a number of time before it moves on.
Arguments	The failure of any component can result in data loss or unreliable output. The system protects against these using redundancy mechanisms.

Table 23: Decision 2: solution for reliability through redundancy

Name	**Solution03: Buffer to cache processes that require farm component**
Decision	#3
Status	Approved
Problem/Issue	Farm Component is not available
Decision	BufferComponent logs status of current process and tries to reestablish the connection with the farm component in order to trigger the process again using the heart-beats.
Alternatives	Retry a number of times and then more on.
Arguments	Buffering data allows temporary storage so that data is not lost even though the farm component is not available at a certain time.

Table 24: Decision 3: solution for failure of farm components

Name	**Solution04: Virtualization of Data**
Decision	#4
Status	Approved
Problem/Issue	Different Sensors might provide different kind of data for the same process
Decision	ViratualizationComponent is a wrapper to prepare Data from different sensors. Because of this data can be treated always in the same manner without the loss of precise computation
Alternatives	The data coming from each sensor can be individually parsed. This requires a high development and maintenance costs.
Arguments	Data virtualization provides an efficient way for data management that allows the application to retrieve and manipulate data coming from different sensors.

Table 25: Decision 4: solution for data virtualization

Name	**Solution05: Inability to perform audio recognition due to noise pollution**
Decision	#5
Status	Approved
Problem/Issue	A farm could be located in an area with a lot of noise polution. This could interfere with the audio recognition algorithms
Decision	During installation a maintainer should measure the level of noise polution. If there is a lot of noise, the farm can be flagged to apply a noise filter as preprocessing method for audio recognition.
Alternatives	Install isolation material between the source of noise and microphones
Arguments	The installation of isolation material will be added on top of installation costs. This makes the product less desirable. If it is possible to solve the issue at the software level, then this is preferred.

Table 26: Decision 5: solution for inability to perform audio recognition

Name	**Solution06: Delay of release due to lack of usability**
Decision	#6
Status	Approved
Problem/Issue	Due to insufficient user feedback during development, usability of the UI components may be insufficient
Decision	During the development process, the UI development team should conduct frequent surveys in order to evaluate the usability of their design.
Alternatives	AB-testing could be done during initial release in order to gather data which can be used to identify possible causes of insufficient usability.

Table 27: Decision 6: preventive measure to prevent delays due to lack of usability

Name	**Solution07: Native or Web UI**
Decision	#7
Status	Approved
Problem/Issue	It is too expensive to develop both
Decision	A browser based UI will be used because avoid cross-platform compatibility issues. With mobile-first development in mind the UI can be made to fit various screen sizes. By supporting the four most common browsers, the vast majority of personal devices can be supported.
Alternatives	Create a native application for each target platform. This would improve usability slightly, since it can be integrated well into the target platform.

Table 28: Decision 7: browser based UI

6 System Architecture

This section provides a high-level description for the Famvolution system explaining the functionality of the system. It starts with an initial model as well as possible alternatives. Then, the chosen Initial Model is developed into an elaborated model with a detailed description. Lastly, verifications are made regarding the feasibility of the elaborated model.

6.1 Initial models

The initial model for the Architecture is detached from the high-level design decisions that are listed in the previous section. Figure 5 shows the actors and the global system components, as well as their logical connections and relations. An in-depth explanation of the actors and their actions is provided in Subsection 6.1.1.

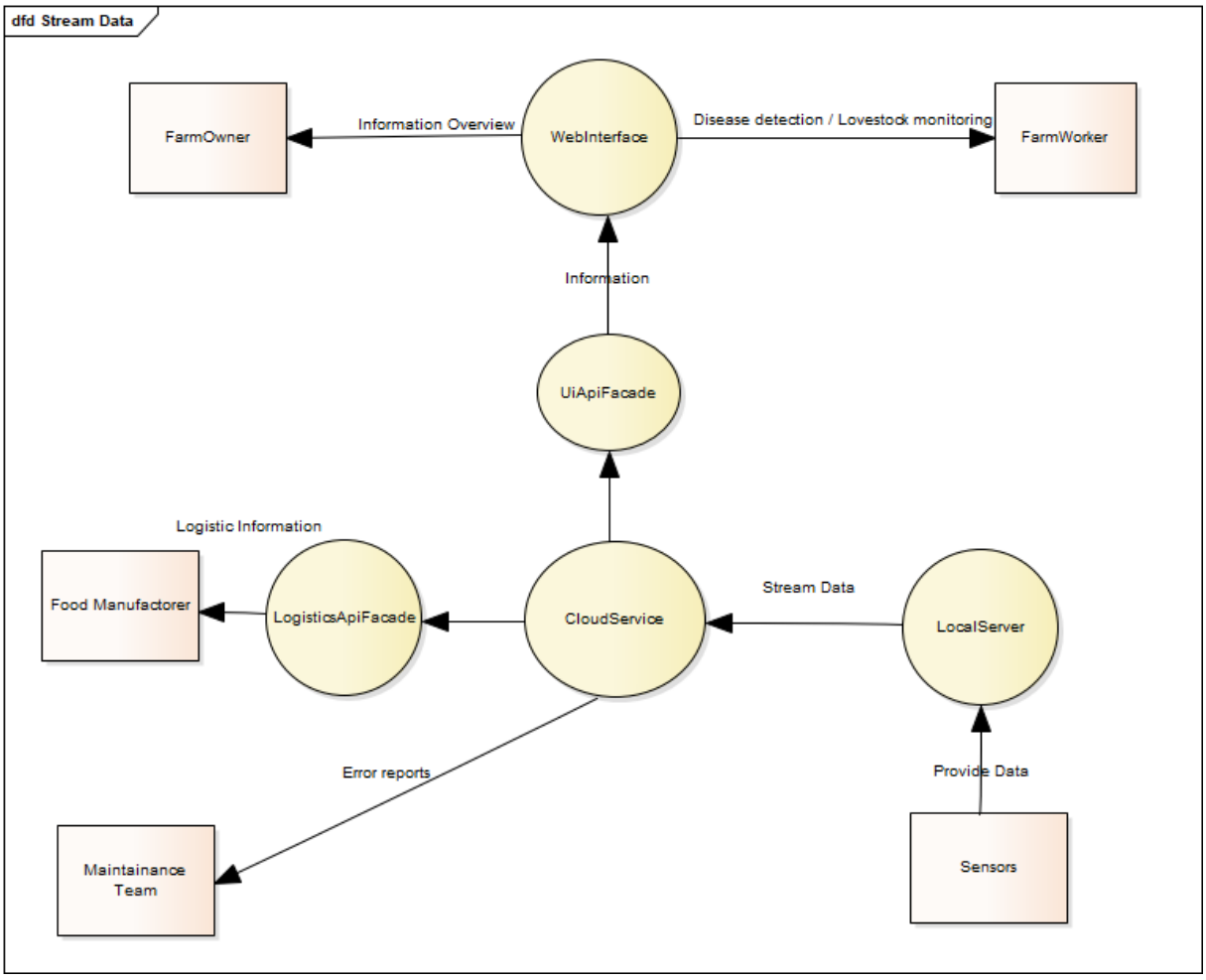

Figure 5: Initial model of Farmvolution

6.1.1 Actors and Roles

Actor is an entity that communicates directly with the system and Role is defined by the Actor's interaction with the system. Below is the list of Actors and the kind of communication they have with the system.

Farm Worker
Farm worker uses the GUI to receive notifications and to retrieve information from the reports generated by the system.

Farm Owner
The Farm Owner can perform Farm Worker's actions and, besides this, can also allow access to food manufacturers and define automated farm processes.

Food Manufacturing Companies
Are the one that purchase products from the farmer. They receive detailed reports and statistics about the production rate of the farm from the cloud services and do not provide any information to the system .

Maintenance Team
Maintenance Team solves problems and are concerned with the status of the components of the system like sensors, cloud platform and local server. They monitor the status of components and act only when something fails or needs upgrades.

6.1.2 Components and External Systems

Following is the list of all the components and systems/sub-systems of external parties that together assemble the entire Farmvolution System as shown in Figure 5.

Cloud Services
This component receives and also stores the sensor data and external data received from the farms. It is used for processing the data and running machine learning algorithms for detection of diseases. The results obtained are then sent to farm workers, farm owner, food manufacturers and maintenance team through APIFacade.

Sensors
This component are the sensors that collect data send it the local server present on the farm.

Local Server
This component is stored locally on the farm and receives data from the sensors. The local server then pre-process the data and streams the data to the cloud services.

Web Interface
This component is the GUI which is accessed by the Farm Workers and Farm Owners.

LogisitcAPIFacade
This component is used to provide a high level abstraction between the cloud services and outsource the information requested to the food manufacturers.

UiAPIFacade
This component is used to provide high level of abstraction between the cloud services and outsource the information related to animals and notification of diseases.

6.2 Elaborated model

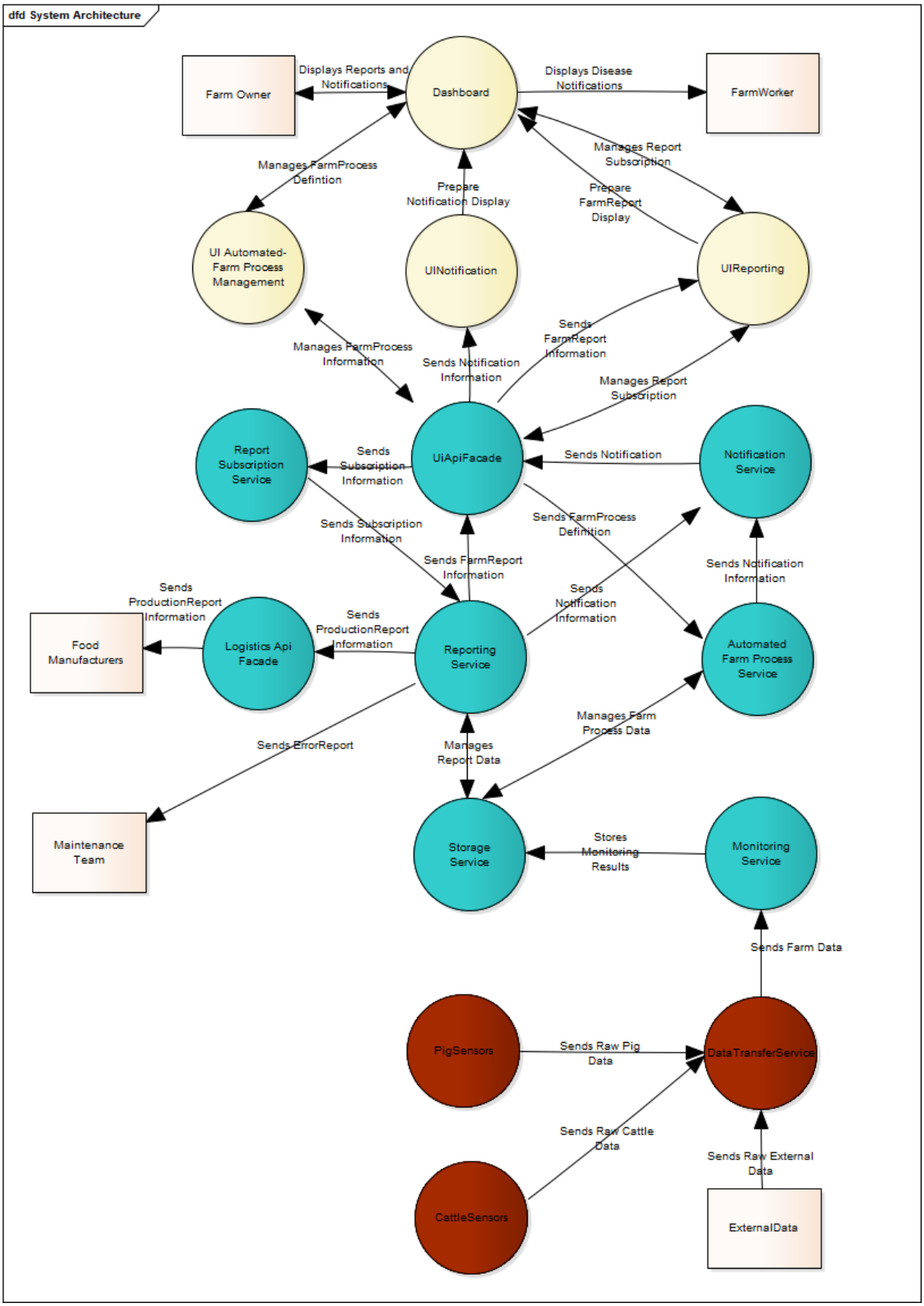

Figure 6: Elaborated model

The initial model as shown in Figure 5 with the given components and communication channels is developed into an elaborated model as shown in Figure 6. The blue color represents the components deployed on cloud services. The red color represents the components present on the farms. The yellow color represents the components of the web UI component. The communication between farm, cloud and web component is carried out using the HTTP protocol. The communication between the cloud and maintenance team is carried out through email. Communication between cloud and food manufacturers is carried over REST Api.

The details of the elaborated model from Figure 6 are as follows:

Pig Sensors
Pig sensors consists of multiple sensors used for collecting data about the audio, food and water intake which is streamed to the local server.

Cattle Sensors
Cattle sensors consists of Thermal cameras used for collecting images about which are sent to the local server.

External Data
This component consist of data recorded by the farms such as the milk produced by each animal, RFID which is integrated with our system through the local server.

Monitoring Service
This component is used for prediction of disease using the machine learning algorithms which receives data from the data transfer service. The results obtained are stored in the storage service.

Storage Service
This component is used storing data from monitoring services and manages data for automated farm process and reporting service.

Reporting Service
This component is used storing data from monitoring services and manages data for automated farm process and reporting service.

Notification Service
This component is used for sending notifications to the UiApifacade. It receives the information related to the disease detected in animals defined by the automated farm process service. The reporting service sends notification information to this components when a report has been generated.

Report Subscription Service
This component receives information related to subscriptions by the farmers which is sent to the reporting service to generate timely report.

Dashboard
This component provides a summary of UIReporting, UINotification and UI Automated farm process management to the farm owner and farm worker.

UIReporting
This componenet is used for reading and writing subcriptions for the reports. It also displays the reports to be viewed by the farm owner and farm worker.

UINotification
This components provides notification when a disease for cattle or pig has been detected.

UI Automated farm process management
This component is used for modification of the automated farm process which can be defined by the farm owner. It can also read and write the farm process information to the UIApiFacade.

6.3 Verification

The feasibility verification of the system architecture is discussed in this section.

6.3.1 Availability

The availability of the system can be defined by considering the Mean Time to Failure (MTTF) of each component in the system, as well as their Mean Time to Repair (MTTR). The availability can be expressed as:

$$\text{Availability} = \text{MTTF}/(\text{MTTF} + \text{MTTR}) \tag{1}$$

Table 29 contains the availability of the Farmvolution's components.

Component	MTTF	MTTR	Availability
Ethernet Cable	3 years	1 day	0.9987
Sensor	3 years	10 hours	0.9916
Local Server	2 years	12 hours	0.9907
Database	3 years	30 minutes	0.9995
Reporting	3 years	30 minutes	0.9985
Monitoring	3 years	30 minutes	0.9970
Disease Monitoring	3 years	30 minutes	0.9900
Web Interface	2 year	30 minutes	0.9947

Table 29: Availability of the system

All components that are located in the cloud are images, numerical data and audio which can be easily restarted and deployed redundantly, therefore the MTTR is estimated quite low with only 30 minutes repair time. Combining all the availability scores in Table 29 result in a global availability of **0.9954**.

6.3.2 Time to Market

All the individual components that will be used to implement the entire system, don't pose a thread for the launch date of the system. All those components are widely used and currently easily available on the market. Therefore the system's launch date entirely relies on the development of the software that will be running within the system. As discussed in section 3.8.1 Software architecture costs and in section 3.8.2 Development cost, the system will be ready for deployment within the period of 48 weeks.

6.3.3 Cost and Return on Investment

The cost for the entire implementation of the system can't be specified clearly, because the scale of the system is different for individual costumers. Some of the farmers might have the need to monitor only the cattle and some farmers might need to monitor only the pigs for their farm. On the other hand there are also farmers that want to monitor both cattle and pigs. Beyond the part of monitoring animals there is also the size of the farm as consideration. Bigger farms need more sensors for more accurate data collection. The software architecture of the system is estimated around €55,000 as described in Section 3.8.1; the development cost is estimated around €15,200 per week as discussed in Section 3.8.2. The initial development is estimated to take about 4 months (up to the end of Q1 2019) or 16 weeks, which amount to € 243,200. The hardware costs for the current packages amount to . The final system cost is then estimated to cost around € 298,200, excluding the hardware costs.

7 Hardware Architecture

In this section hardware architecture will be analyzed in depth. The section first describes the hardware design decisions. Therefore a description is provided of the hardware architecture. The section ends with an overview of the application interface.

7.1 Hardware design decisions

In order to determine the hardware components, there are several decisions to be made. The hardware that have been decided to be used for the system implementation are listed as follows:

Decision HD-1	**Local Data Storage**
Problem / Issue	Fast hard drive for local server
Status	Approved
Decision	The system will use Samsung 860 Evo 500GB SSD
Alternatives	WD Blue 3D 2TB, SanDisk Ultra 3D 2TB, Crucial MX500 2TB and many more. Also any HDD could be used for local storage.
Arguments	The harddrive on which the operating system and all software is installed should be fast, so SSD is prefered over and HDD. Therfore the best one in market is a Samsung 860 Evo 500GB because it is one of the fastest one in the market and reasonably prices. With speed up to 441MB/s(write). A capacity of 500GB is sufficient for the basic local server.

Table 30: Decision HD1: Local Data Storage

Decision HD-2	**Local Data Storage Extension**
Problem / Issue	High volume local storage for pigs
Status	Approved
Decision	Seagate SkyHawk 10TB 3,5 inch
Alternatives	Seagate HDD 3.5” 10TB, WD Blue 3D 2TB, SanDisk Ultra 3D 2TB, Crucial MX500 2TB and many more. Also any HDD could be used for local storage.
Arguments	An SSD is not an option due to the high volume of data that is being written every day. The lifespan of an SSD suffers from this, since a bit can only be rewritten a finite amount of time. An HDD does not have this constraint. The Seagate SkyHawk is optimized for storing video data and its high capacity of 10TB makes it a perfect fit.

Table 31: Decision HD1: Local Data Storage

7.2 Description of Hardware Platform

All the sensors and the data collection methods are operated within the farm itself. To increase fault tolerance, sensors are placed on a location that is protected from weather, noise pollution and other environmental factors. Sensors communicate with the system via Ethernet cables to make sure that the data will be transmitted safely and reliably to the system’s processing unit (Local Server).

Decision HD-3	**Thermal Cameras**
Problem / Issue	Cameras that detects temperature of cows
Status	Approved
Decision	The system will use FLIR A310 thermal cameras
Alternatives	N/A
Arguments	These cameras are developed by Swedish company for the exact same purpose. They are the only cameras available on the market that have been tested for this kind of work.

Table 32: Decision HD3: Thermal Cameras

Decision HD-4	**Kinect V2**
Problem / Issue	Record video, depth and audio for pigs
Status	Approved
Decision	The system will use KinectV2 camera
Alternatives	Rode VideoMicro, Aputure A.lav Lavalier microfoon, Rode Videomic GO, Rode Videomic Me
Arguments	The Kinect V2 is relatively cheap and it has the ability to detect surround sound because it has an array of 4 omnidirectional microphones. The sensitivity of this microphone can reach up to -42 dB. The other advantage is that it also has video and depth sensing capabilities.

Table 33: Decision HD4: Kinect V2

There will be two main platforms that measure weight of animals and detect potential diseases. These consist of:

1. Two thermal cameras for cows
2. One kinect v2 camera for pigs

The Kinect camera will be placed inside the pigs' stable in order to monitor the sound of pigs in real time and detect any suspicious anomalies in the sound, like coughs. The Kinect will also record video and depth data of the pigs that can be used for object tracking and monitoring feeding behaviour.

All collected data will be pre-processed at a bridge and travel through a local server where it is compressed before reaching the cloud. On a local server, the collected data will be buffered for up to 24 hours when the cloud becomes unavailable due to connectivity issues. Long term data storage happens in the cloud so that it is easily accessible from anywhere.

7.3 System Interfaces

The system's interface is mostly the interaction of user using web based applications that can run on any device. The easy to use GUI will be easily used by any technology inexperienced farmer. The application will communicate with the local server and the server will enable the appropriate actuators. The actuators are connected with the hub using cables to prevent signal distortion.

7.4 Hardware Overview

In this section the specifications of the hardware components that are used will be outline. The decisions that were taken in consideration of choosing those components can be found in Description of Hardware Platform.

Figure 7 shows the hardware overview of the system where the sensors used for collecting pig data is the Kinect V2 and for collecting cattle data is FLIR A310 Thermal camera. These sensors are connected to a Raspberry Pi which is a bridge to send the collected data to the local sever. The local sever consists of a network card for receiving the data and hard drive for storage and powerful CPU.

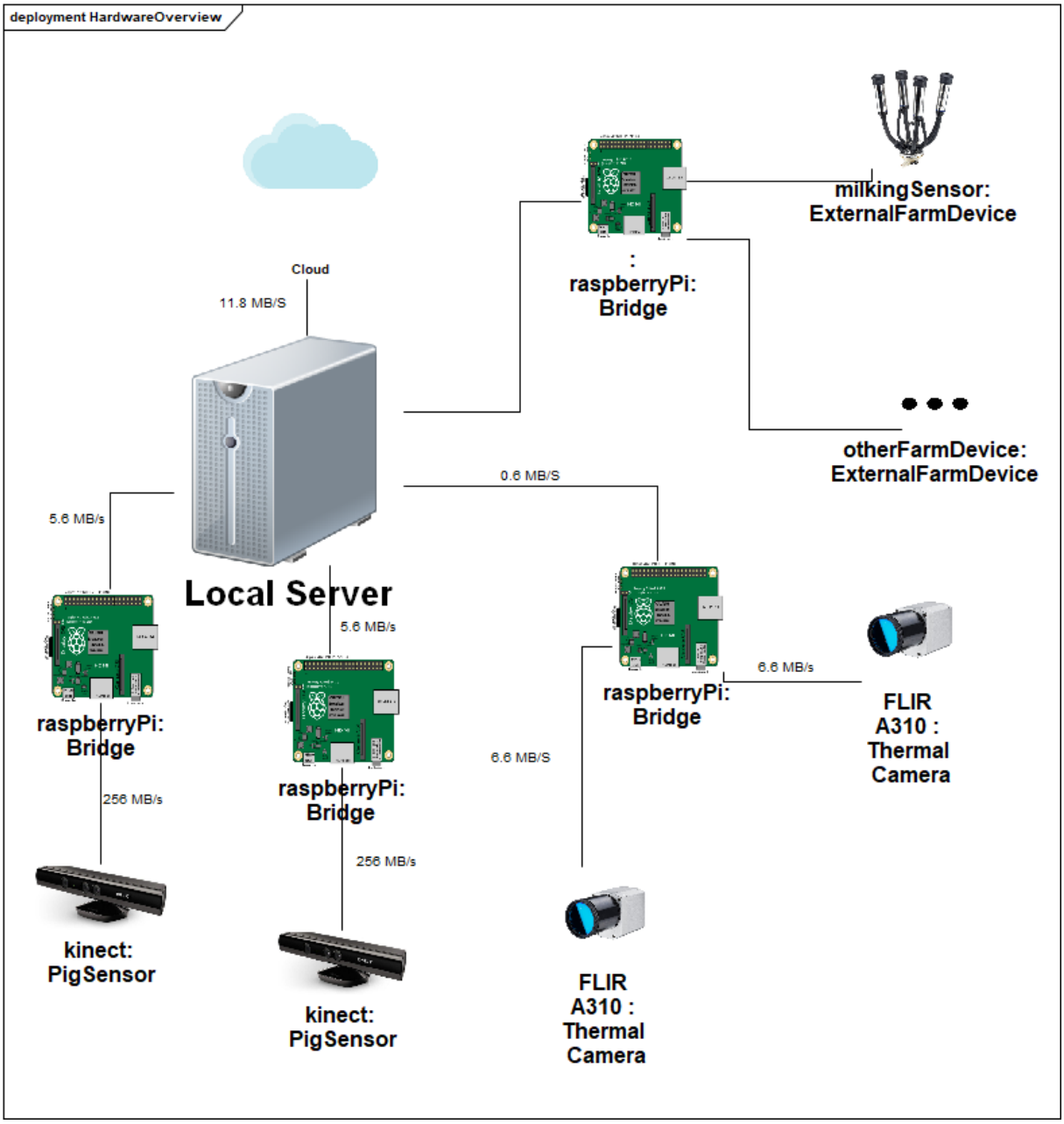

Figure 7: Hardware Overview

7.5 Server Specifications

In order to determine reliable hardware specifications for the server, we must first describe in detail which data must be handled by the system and what the volume of that data is. As was mentioned in earlier sections, there are two types of sensors arrays that collect data for Farmvolution:

1. Kinect v2 cameras for pig monitoring
2. 2 A310 FLIR cameras and one RFID sensor for cow monitoring

7.5.1 Network Specifications

The kinect v2 camera consists of one GBRA camera, one RGB depth-sensor and four microphones that take 32 bit samples. The GBRA camera records a resolution of 1920x1080 pixels at a maximum of 30 frames per second (fps). The depth-sensor records 512x424 pixels at 30 fps. Each microphone has a sample rate of 16000 HZ. and a single sample is 32 bits. Using these numbers, we arrive at the following peak raw data collection rate for a single pig setup (10-15 animals per Kinect v2):

1. Camera: $(1920 * 1080) * 30 * 32 = 1990656000 bits/s = 237.3 MB/s$
2. Depth sensor: $(512 * 424) * 30 * 32 = 156303360 bits/s = 18.6 MB/s$
3. Audio: $32 * 16000 * 4 = 204800 bits/s = 0.25 MB/s$
4. Total: $237.3 + 18.6 + 0.25 = 256,15 MB/s$ of raw data collection.

Handling this rate of data transfer on the local network from sensor to local server is possible using high end network cables, but not desirable. Considering that multiple sensor arrays will be collecting data at a single farm, transmitting the data to the CloudComponent without compression is impossible using conventional networking solutions. Therefore, we decide to use a two-stage compression algorithm explained in Section 8.2.1.

The first compression step is performed at the bridge between Kinect and Local Server and can be seen as a transformation into an intermediate form that is reduced in size, but still capable of being used in computations. This step aims to reduce the amount of data to a manageable level for transmission through Ethernet cables present at the farm. The second compression step is done on the Local Server, where data is compressed using a conventional algorithm for storage and transmission to CloudComponent. In this stage, data must first be decompressed before additional processing can be done.

The FLIR A310 cameras record at 30 FPS with a resolution of 320x240 pixels in RGB format. The livestock scale and FLIR camera both measure a single number per cow. Even with added overhead for data transfer, this data volume is negligible compared to that of the FLIR camera. According to this specification, the volume of raw data collected in a second would be:
$320 * 240 * 30 * 24 = 55296000$ bits/s = 6.6MB/s

7.5.2 Processing Specifications

The first compression step will be handled by the RaspberryPi bridge. This is where images are scaled, the frame difference is computed and a point-cloud model is constructed. The scaling of the RGB image reduces the raw data by a factor of 2.39 for GBRA data. Application of the frame difference method yields another reduction by a factor of 22 for camera and depth data. On the local server the LZ4 library is able to compress at a rate of 2.073.

This means that a total compression ratio of $2.39 * 2.073 * 22 = 109$ is possible for the camera. For the depth sensor a compression ratio of $2.073 * 22 = 45.6$ for the depth sensor. Combining this with raw data collection rates, the following data transfer needs have been determined for a single Kinet v2 camera. The first list shows the calculation of data transfer rates from bridge to local server. The second list shows the transfer rate from local server to the CloudComponent.

Data transfer from bridge to local server:

1. Camera: $237.3MB/52.6 = 4.511$ MB/s
2. Depth sensor: $18.6MB/22 = 0.845$ MB/s
3. Audio: 0.25 MB/s
4. Total: $4.511 + 0.845 + 0.25 = 5.6069$ MB/s of preprocessed data to local server.

Data transfer from local server to CloudComponent:

1. Camera: $237.3MB/109 = 2.177MB/s$
2. Depth sensor: $18.6MB/45.6 = 0.408MB/s$
3. Audio: 0.25 MB/s
4. Total: $2.117 + 0.408 + 0.25 = 2,775$ MB/s of compressed data to CloudComponent.

The final compression step will be done on the local server using the LZ4 compression library. An intel Xeon-E3 v6 is able to compress around 1000MB/s. This is based on a 730MB/s benchmark for an i7-6700K[20] combined with the higher base frequency of the Xeon CPU. This is roughly equal to the amount of data gathered by 175 Kinect v2 cameras and therefore sufficient for the average farm.

7.5.3 Storage Specifications

According to NFR1-2 the system must be able to buffer up to 24 hours worth of compressed sensor data. Using the data collection rates that we just calculated, the required amount of storage a single kinect camera would be:

$$86400 \text{ s} * 2.775 \text{ MB/s} = 234 \text{ GB per kinect camera}$$

For a single cow on average 3 seconds and at most 10 seconds of thermal imaging will be recorded per milking session. There are two milking sessions per day. Therefore, the storage needed in the worst case is:

$$6.6 \text{ MB/s} * 10 \text{ s} * 2 = 132 \text{ MB per cow}$$

Figure 7 also illustrates the data transfer rates, processing and storage needs for Farmvolution based on the calculations listed above.

[20] https://github.com/lz4/lz4

8 Software Architecture

This section aims to outline the software architecture. Following the Attribute-Driven Design (ADD)[1], it starts with mentioning the Architectural Significant Drivers(ASD). After this the different views following the 4 + 1 [6] view model will be presented.

8.1 Software Architecture Design

To describe the Software Architecture following ADD it is crucial to choose the Components defined in Chapter 6 to refine them. An important further step of is to choose the relevant ASD to select tactics and patterns which are necessary for the refinement.

8.1.1 Relevant Components

The relevant components are Farm Component, Cloud Component and Web Component. Figure 8 provides an overview of the three main components of the Farmvolution system.

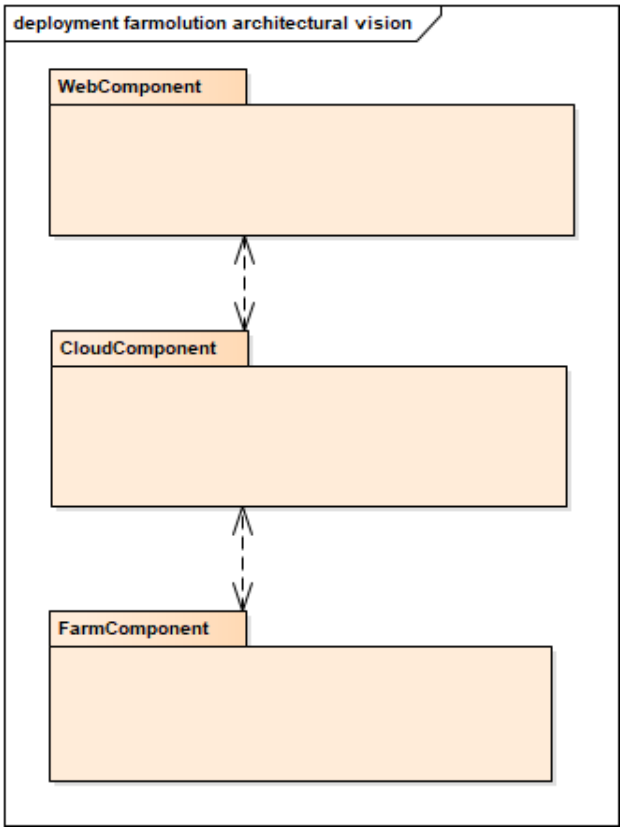

Figure 8: Component overview of Farmvolution system

8.1.2 Architectural Significant Drivers - ASD

The ASD of the Farmvolution System are the Key Drivers (Availability, Performance and Reliability) (cf. Section 4.3), the Use-Cases (cf. Section 4.5) and a selection of the FR's (cf. Section 4.6) and Nonfunctional Requirements (cf. Section 4.7). Table 34 gives an overview on these requirements.

Type	Requirement
FR	FR1, FR2, FR3, FR4, FR5, FR6, FR7, FR8, FR9, FR10, FR11, FR12, FR13
NFR	NFR0-2, NFR0-3, NFR0-4, NFR0-5, NFR0-6, NFR1-1, NFR1-2, NFR1-3, NFR2-1, NFR4-1, NFR5-1,

Table 34: Architectural Significant requirements of Farmvolution

8.2 Architectural views

This subsection provides an overview on the architectural views. It follows the 4 + 1 view model including logical view, implementation view, process view, deployment view and use case view.

8.2.1 Logical view

This view shows the structural elements of the Farmvolution system. It also gives an overview on its logical organization, the allocation of functionality and the behavior. In order to describe the logical organization a description of the three main components is provided. After this the connection between the structure and the behavior is illuminated.

Logical Organization
The Farmvolution System is structured by the Component Oriented Software Architecture (COSA) approach. This approach has been chosen in order to encapsulate functionality and follows the separation of concern principle. In addition, this architectural design enables the communication between the certain subsystems namly WebComponent, CloudComponent and Farm component back and forth. A similar and often used approach of the 3-Layer-Architecture is not feasible here since e.g the FarmComponent needs to be able to demand certain functionality provided by the CloudComponent. Such a communication is not allowed by the 3-Layer-Architecture.

Figure 8 gives an overview on the three main components. The following passage will provide a short description.

WebComponent
The WebComponents enables an entry point to the user. Basically, it is a Dashboard where farm owner or farm worker can see farm reports, see results of health monitoring, subscribe for particular reporting information and change settings. The WebComponent is responsible to deliver the web interface to the user's browser. Its services have access to the CloudComponent via an UiApiFAcade the CloudComponent provides.

CloudComponent
The CloudComponent is the component that enables calculation, access to third party tools and storage. It provides several ApiFacades for communication with WebComponent, Farmcomponent, third party tools that provides analyzing features for e.g. health monitoring and food distributers.

FarmComponent

FarmComponent is the subsystem that is locally installed at the farm. It collects the data from all sensors, triggers actors, process orders coming from CloudComponent and requests functionality from CloudComponent.

Structure of FarmComponent

Crucial key drivers of Farmvolution are availability and reliability. On the one hand that requires tactics to ensure that data of the FarmComponent is available and on the other hand that this data is consistent. One possible event that could interfere with this demand is a failure of the FarmComponent. In this case data can get lost. After restoring the system, the system cannot process this data anymore and hence the information is neither available nor reliable. Another scenario might a failure of CloudComponent during processing the data delivered by FarmComponent. The data has to be stored in the FarmComponent during this phase until all processes are finished and reliable output is stored in the CloudStorage. To enable this a system wide transaction mechanism is required that is capable of tracking the state of the data. The Audit Trail Pattern provides a solution to verify that these transactions are processed correctly and honestly [5]. Applying this pattern to the FarmComponent enables tracking of the current state of the data and whether the data has to be stored locally. The Farm component has several nodes of devices that send data back and forth (see Figure 7). To ensure that no data is lost because of a failure of either FarmComponent or CloudComponent, each time a device node receives a data set it will store it locally until it receives the message that the next component has received it and finished all required processes. Thus, the state of all data can be restored after one part of the system fails during collecting, sending or processing data.

Figure 9 provides an overview on FarmComponent's software design. The transaction tactic mentioned above, is reflected by TransactionManager and TransactionProcessor interface. The TransactionProcessor saves the state of a Snapshot. Snapshot acts as a Wrapper for the four classes Data, Command, Response, and Request. Each time one of those objects is created the TransactionProcessor sends it to the LocalStorageService for storage and creates an Transaction. This Transaction is a part of a TransactionRecord which is managed by the TransactionManager interface. The TransactionManager supervises all services that are responsible for the processes of the FarmComponent, namely creatíng and sending data, processing commands from the CloudComponent, and managing communication with the CloudComponent. Each time one further step within this transaction chain was made the TransactionManager records this in the particular TransactionRecord (see Figure 20). Thus, the FarmComponent is able to track the different states of its data flow and can manage to persist all crucial information. One final comment on avoiding loss of data that is lost in the CloudComponent. If the CloudComponent fails during the computation of received data, those data will be lost in the CloudComponent. Extending the transaction chain to the CloudComponent will enable to store specific data in the FarmComponent until it receives the TransactionRecord from the CloudComponent that gives proof of the sucessful computation and storage of the particular information and can thus be deleted in the local storage.

Another crucial part of the FarmComponent is the BufferService. This Service manages requests and responses to and from the CloudComponent. The BufferService receives an Heardbeat signal from the CloudComponent to supervise whether the CloudComponetn is available or not. Since requests and responses are also reflected in the transaction chain, the BufferService is in combination with the Headbeat signal able to supervise which requests or responses has already been received or has been lost.

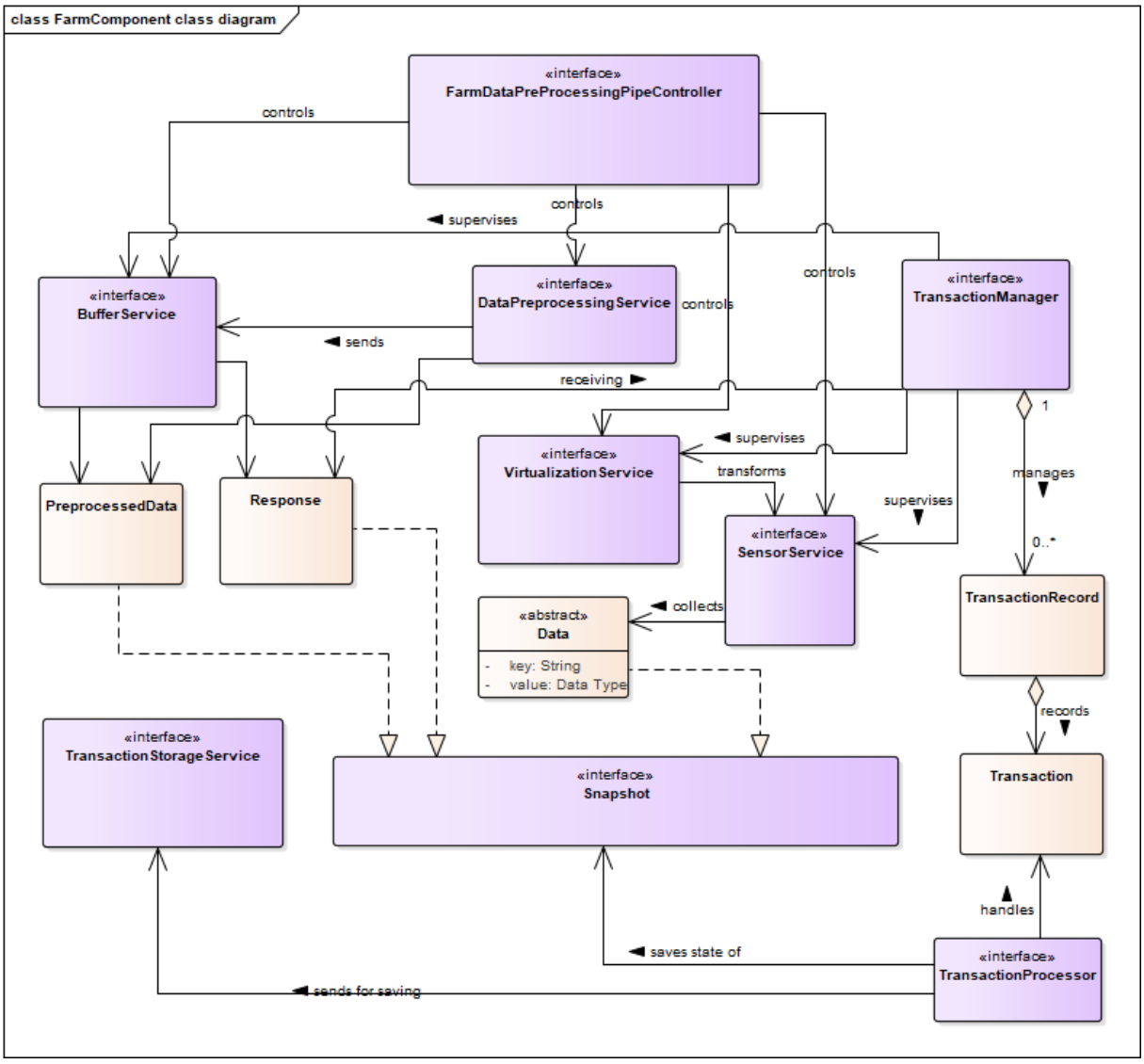

Figure 9: FarmComponent class diagram

Compression algorithm of FarmComponent

The compression algorithm that Farmvolution will use for Kinect data consists of two steps. The first step is executed on the bridge between kinect and Local Server, while the second is executed on the Local Server itself. The algorithm works as follows:

Preprocessing algorithm steps

1. 4 frames are given: `depth[t-1], depth[t], rgb[t-1], rgb[t]`
2. Scale `rgb[t]` so that `scale(rgb[t]) == 2 * scale(depth[t])`
3. Compute the frame difference: `depth[t] - depth[t-1]` and `rgb[t] - rgb[t-1]`.
   Note that a 1x1 area in the depth image maps to a 2x2 area in the color image.
4. Determine the points of interest: select all pixels from both images where at least one of the two differences is greater than 0:
   `|depth_difference[x, y]|` > 0 or
   `|rgb_difference[x, y]|` > 0.
5. Compute the point-cloud representation of the points of interest in the depth frame difference.
6. Annotate each point in the cloud with the corresponding 2x2 RGB pixels.
7. Merge adjacent points that have an equal depth value, combining their RGB pixels into a larger sub-image
8. Transfer point-cloud data to the Local Server
9. Apply a generic compression algorithm from the LZ4 library [7] before storing it in the buffer.
10. Transmit compressed data to the CloudComponent.
11. Decompress using LZ4.
12. Reconstruct the original depth and RGB images from the annotated point-cloud.

Structure of CloudComponent

The CloudComponent is the component that enables all functionallity of the Farmvolution System. I therefore has a bulk of services that interact with each other and process the data and requests coming from WebComponent and Farmcomponent. The key functionallity and its reation to the other services within the CloudComponent will be presented in this subsection. Figure 10 provides an overview of the logical elements of the CloudComponent.

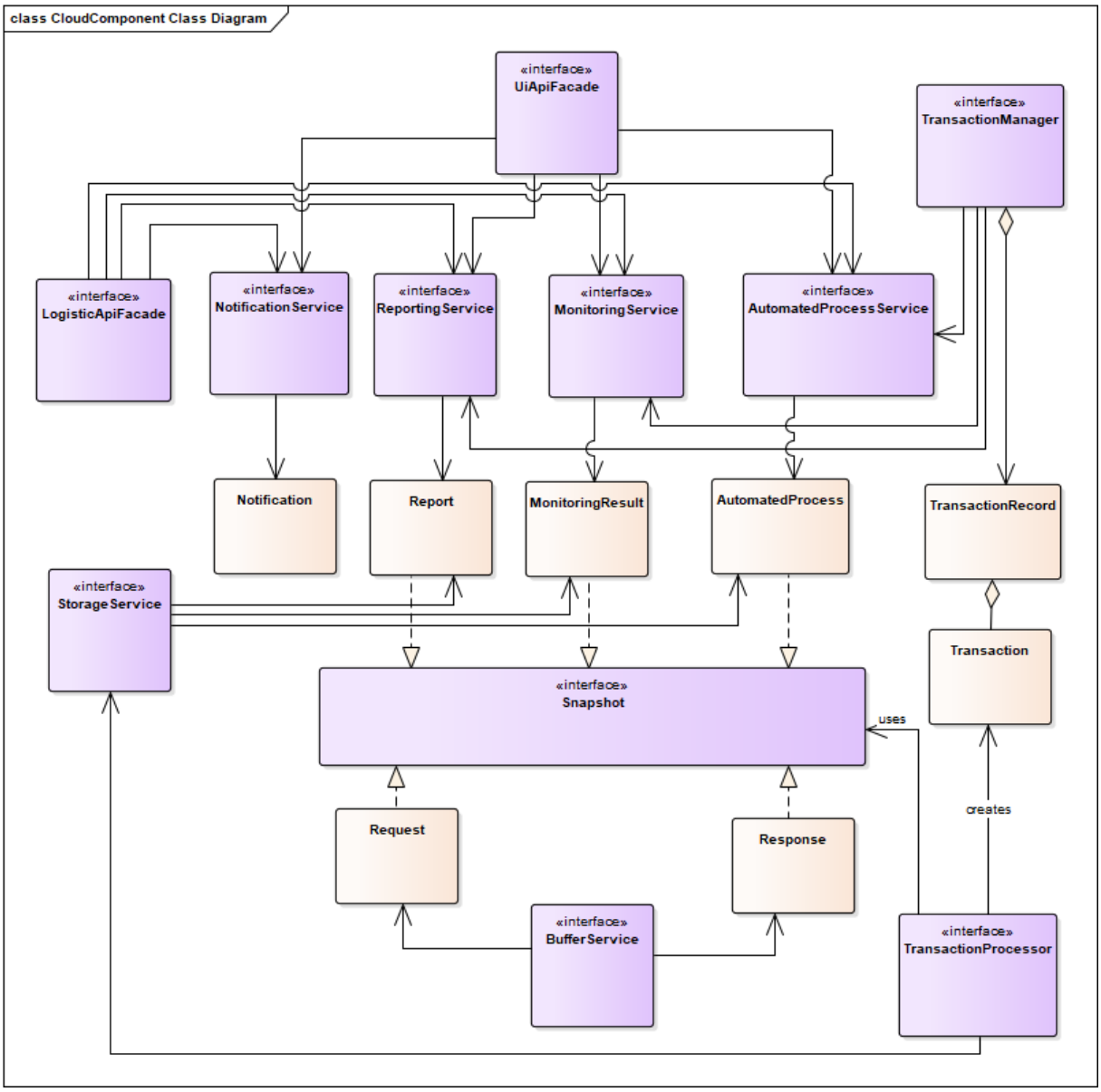

Figure 10: CloudComponent class diagram

To cover the same problems with the key driver reliability, the CloudComponent has a Buffer and a Transaction mechanism. Thus, the transaction chain can be extended from the FarmComponent to the CloudComponent. If both components cannot communicate with each other all data will be stored at the particular location they are at the time of the disconnection. The FarmComponent will hence not clear its storage until it receives a TransactionRecord from the CloudComponent that ensures that the already send data was processed and stored successfully. The Buffer will also ensure that no Requests or Responses will get lost in case of a lost connection.

Another important mechanism for reliable and available data is the Storage service. This service coordinates and manages the redundant database instances and ensures that all data is stored properly.

The functionality of the Farmvolution system is provided by the AutomatedProcessService, MonitoringService, ReportingService and NotificationService. For the communication to the WebComponent as well as to the food manufacturers, the CloudComponent provides two Facades to handle requests and direct them to the correct place. Those are UiApiFacade for the requests coming from WebComponent and LogisticApifacade to manage requests from the food manufacturers.

Structure of WebComponent
The WebComponent is basically a component that is responsible for the UI creation and interaction with the CloudComponent. Since it's structure depends fully on the certain UI-Framework (such as Angular) there is not mutch to show at the moment. In addition, no decision has been made yet on which technology the UI should rely on.

8.2.2 Implementation view

The implementation view present an overview of how the system will be implemented. This is basically which functionality is encapsulated in which component and how the connection between those is. Components can communicate with each other with required or provided interfaces provided by ports of a particular component. This section will provide an overview how this is done for the main parts of the Farmvolution system.

CloudComponent: Figure 11 gives an overview on how the components in the CloudComponent are connect to each other. These connections show how the communication between those components are achieved. The CloudComponent itself has four provided interfaces to grand access to its functionality to external components. Those can be components that belongs to the Farmvolution system as well (FarmComponent or WebComponent), or components that represent third party systems such as systems of the food manufacturers.

Within the component there are several sub-components for the several functions and sub-functions of the CloudComponent. Each of them provide or require interfaces to communicate with other components. These interfaces are attached to ports, which are actually define "interfaces for components". The ports are connected to other ports in that manner that always a provided interface is connected to a required one. The single line between two ports represents the connector that connects these ports.

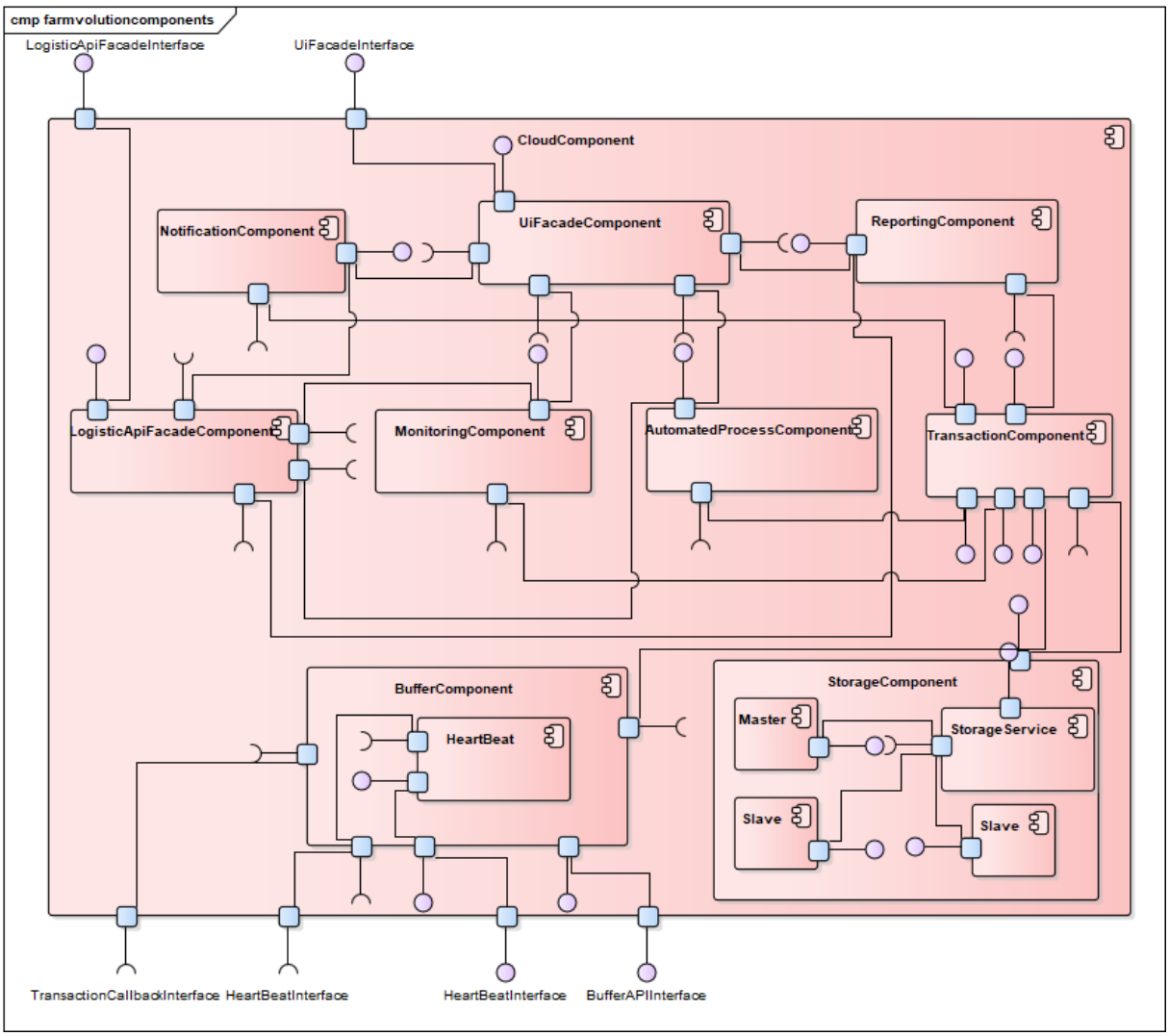

Figure 11: CloudComponent component diagram

FarmComponent: Figure 12 gives an overview on how the components in the FarmComponent are connect to each other. FarmComponent provides an interface to check its heart beat and one to enable callbacks. Inside FarmComponent are several sub-components that enable virtualization of sensor data, transaction tracing, and storage.

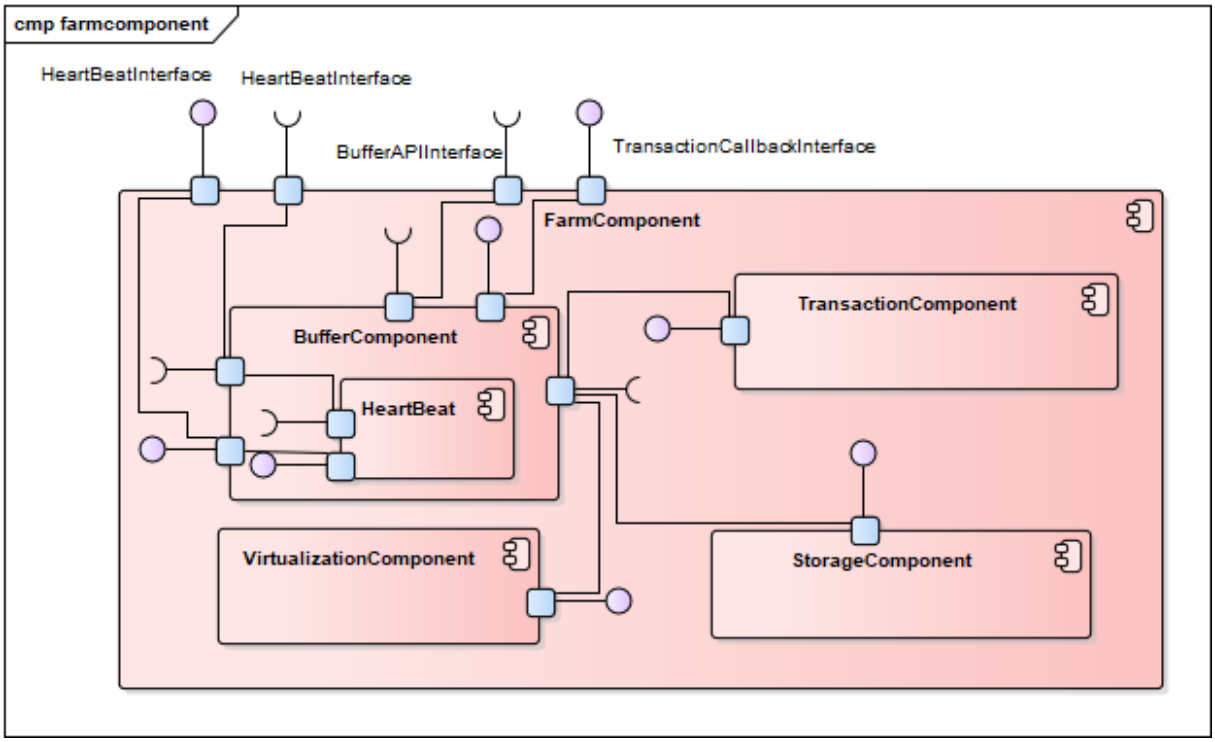

Figure 12: FarmComponent component diagram

8.2.3 Process view

The process view shows how the different functions and sub-functions of the Farmvolution system are working. This section will provide activities diagrams that show the different steps of the behavior of a certain functionality.

Health Monitoring: Figure 13 shows how Farmvolution collects data at the FarmComponent and transport it to the CloudComponent in order to process the data for health monitoring of animals. The process is generalized in order to describe both the health monitoring of cows and pigs. A crucial aspect is the checking of heart-beat. If there is a heart-beat form CloudComponent the data can be transmitted.

Report creation: Figure 14 shows how Farmvolution uses the processed data in order to create a report. After a report was created, it is stored in the cloud storage. In addition, if a farmer has subscribed to a particular report, this process will detect the subscription and notify the farmer.

Drinking and feeding behavior: Figure 15 shows how Farmvolution collects data at the FarmComponent and transport it to the CloudComponent in order to process the data for supervising of drinking behavior. Same as in Figure 13 there is a checking for heart-beat to determine if the CloudComponent is available.

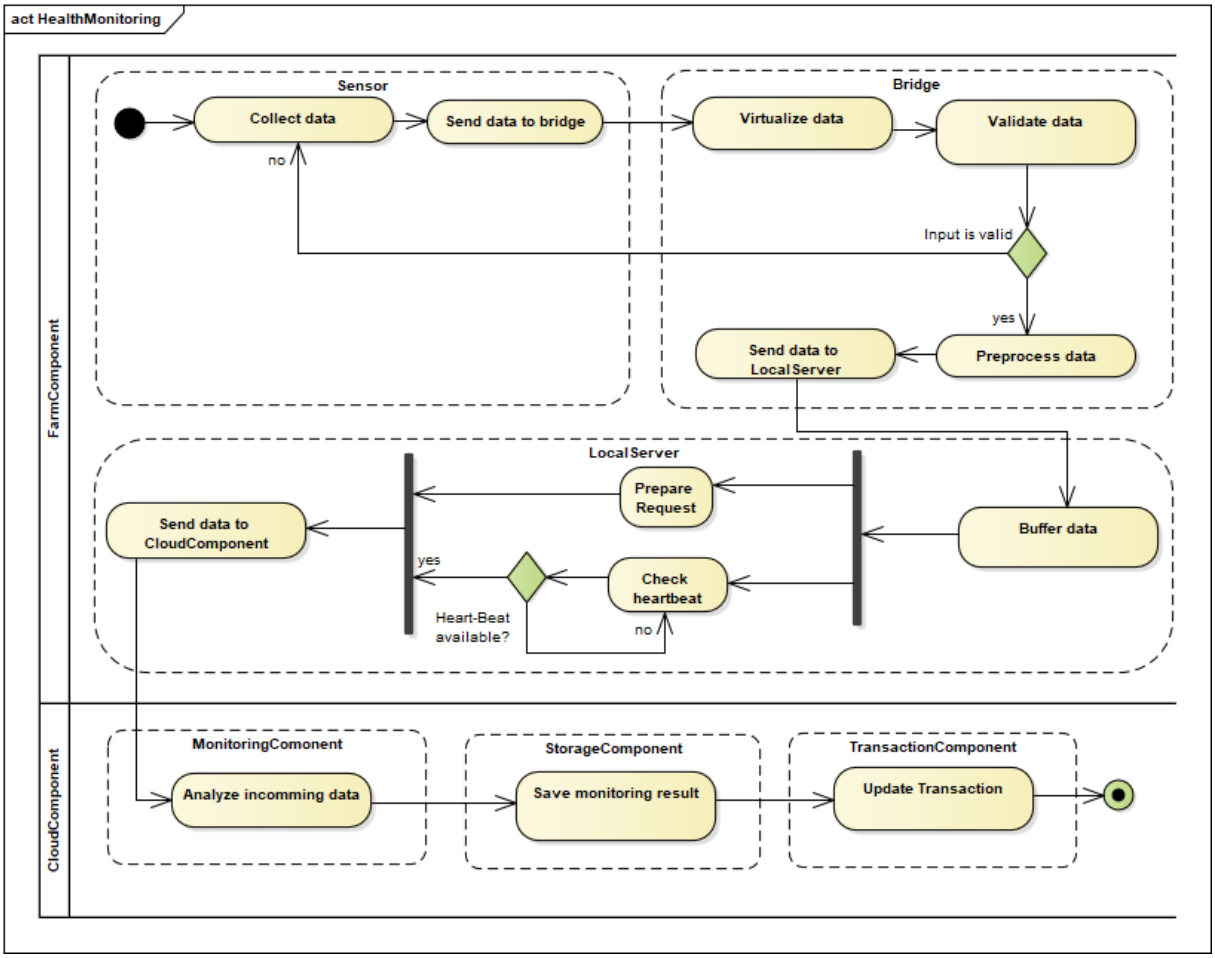

Figure 13: Activity diagram for health monitoring

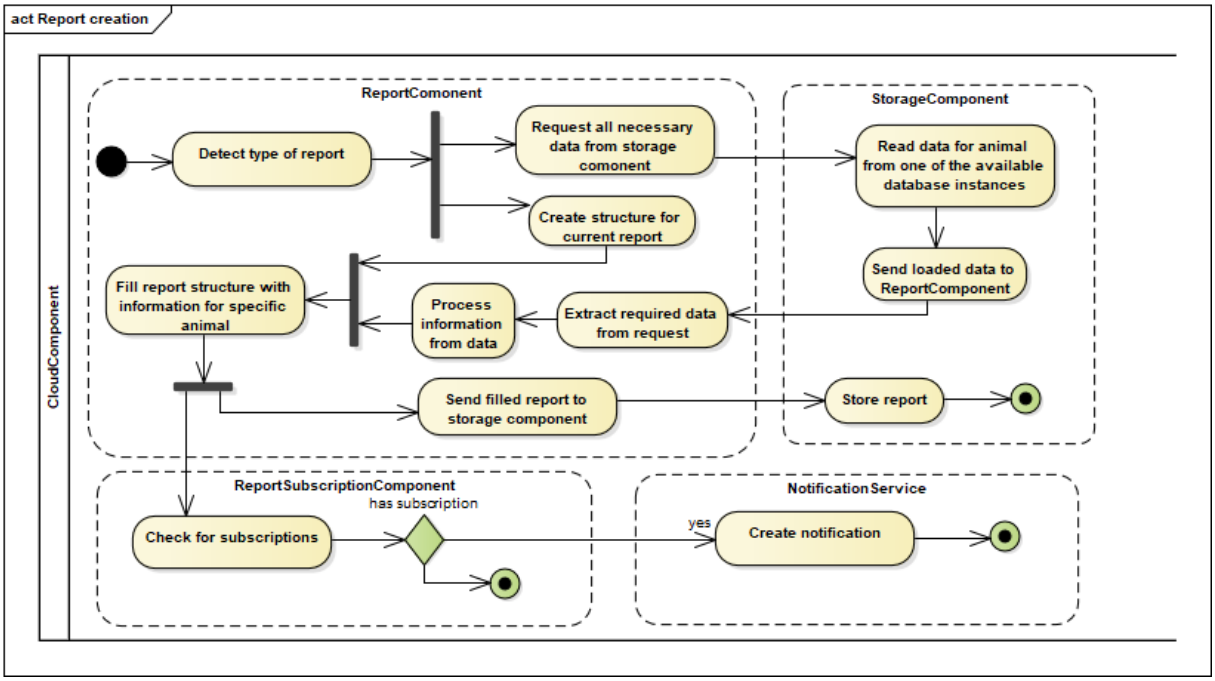

Figure 14: Activity diagram for creating a report

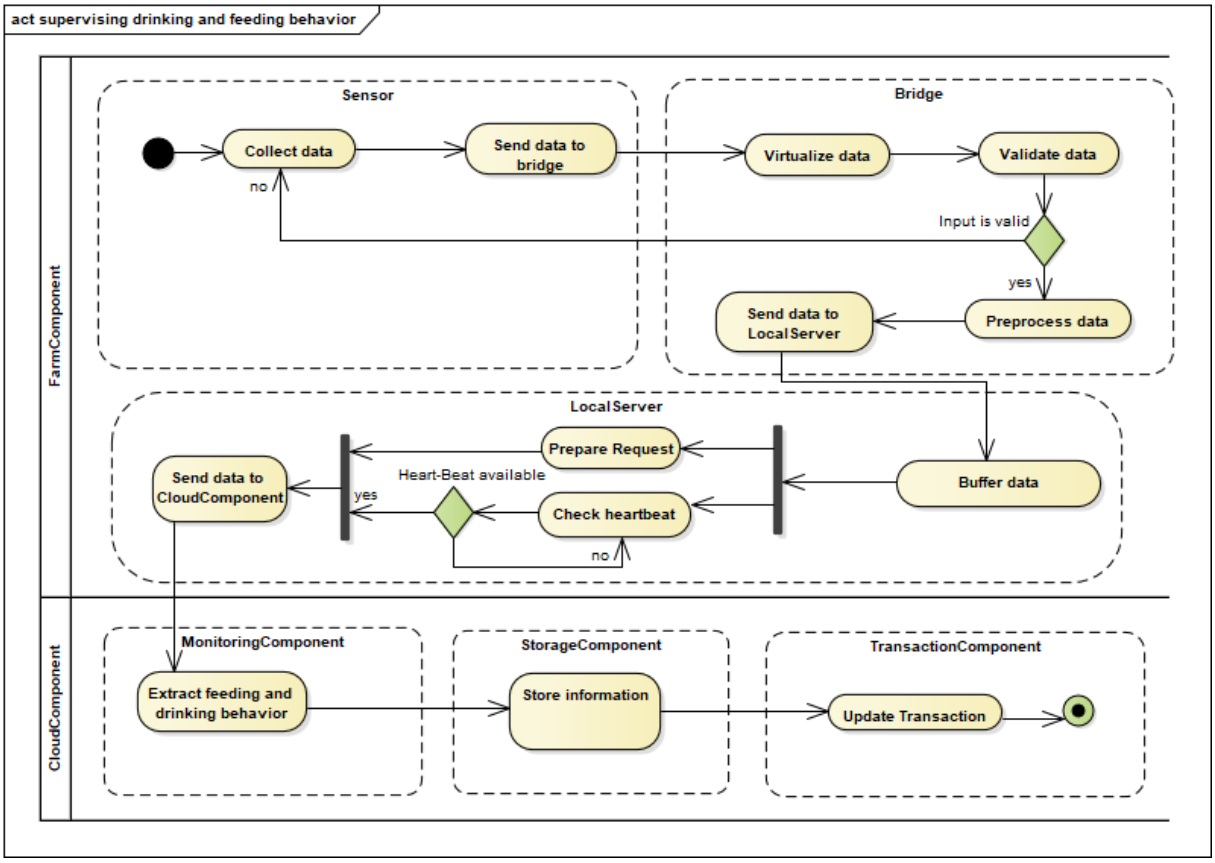

Figure 15: Activity diagram for supervising drinking and feeding behavior

Retrieve farm logistic demands: Figure 16 shows the process to retrieve logitics demands made by the food manufacturer. The food manufacturing system makes a request for the report which is received and processed in the cloud component.

Subscription to report: Figure 17 shows the process for subscription of the report. The reports are selected and subscribed in the dashboard which is then validated in the UIReporting and sent to the cloud component to create a report subscription.

Define automated farm process: Figure 18 shows the diagram for defining automated farm process. The process definitions are extracted and validated in the web component and sent to the cloud component. The cloud component processes this request and checks for the parameters of the process and creates the farm process.

Redundancy in storage: Figure 19 shows the sequence diagram for redundancy in storage. The storage service writes the data to the master database which is then replicated in both the slaves. The storage service can then read the data from any of the two slave database.

Audit trail: Figure 20 shows the sequence diagram for audit trail pattern. The service created a snapshot for any data which then informs the transcationProcessor about the generation of snapshot. Transaction processor creates a traanscation and which is then stored by the transaction processor in the the local storage service. The local storage service add the created transaction in the transaction record supervised the TranscationManager and provides updates to the TranscationRecord. The transaction manager also supervises the Service components.

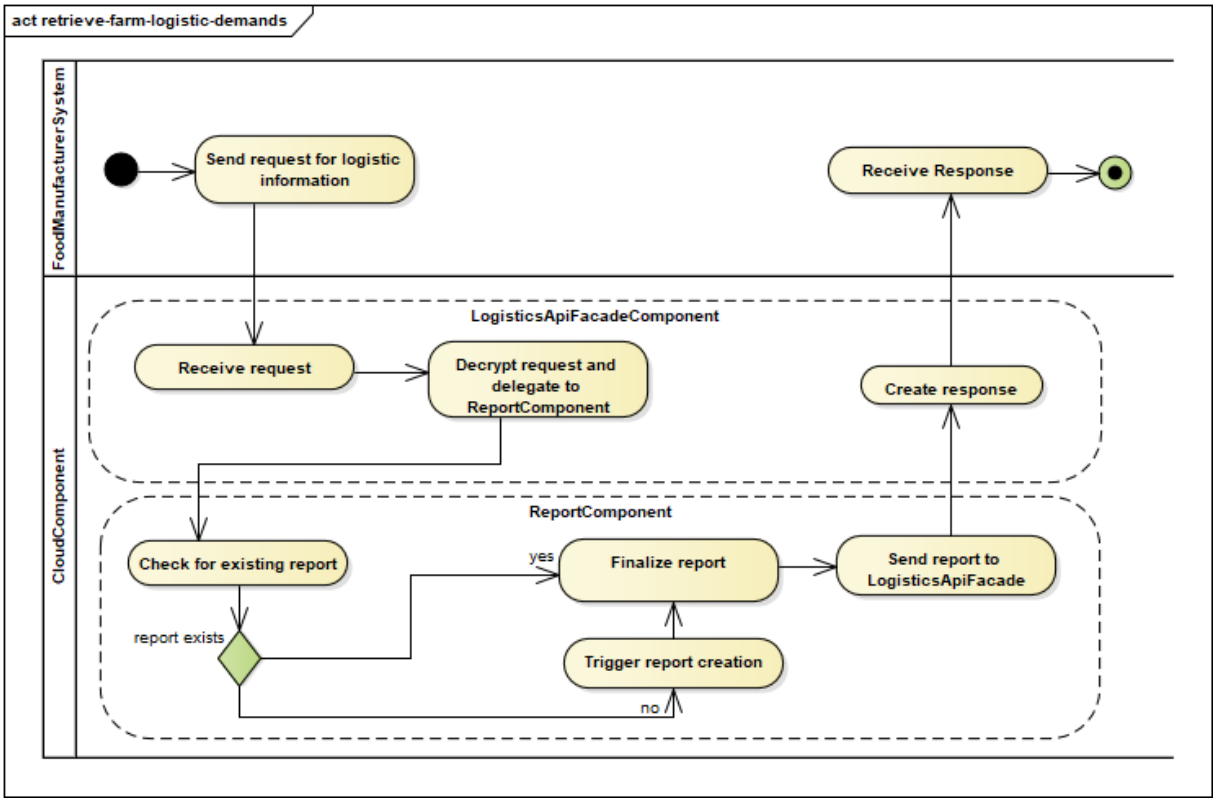

Figure 16: Activity diagram for Receiving logistic demands

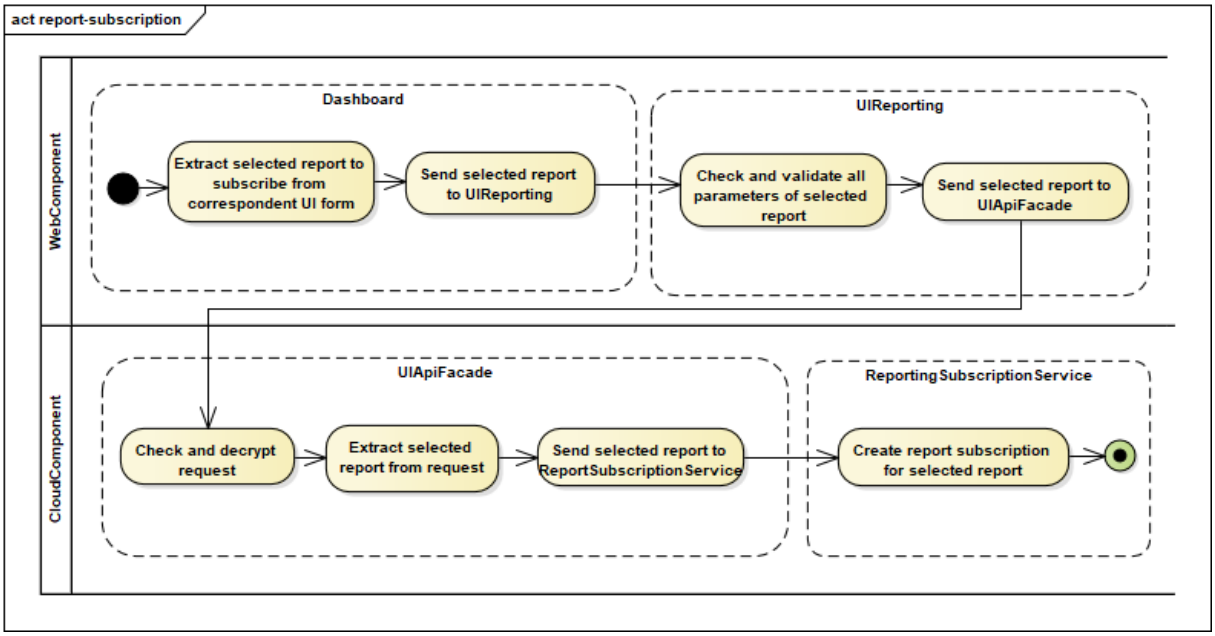

Figure 17: Activity diagram for subscribing for report

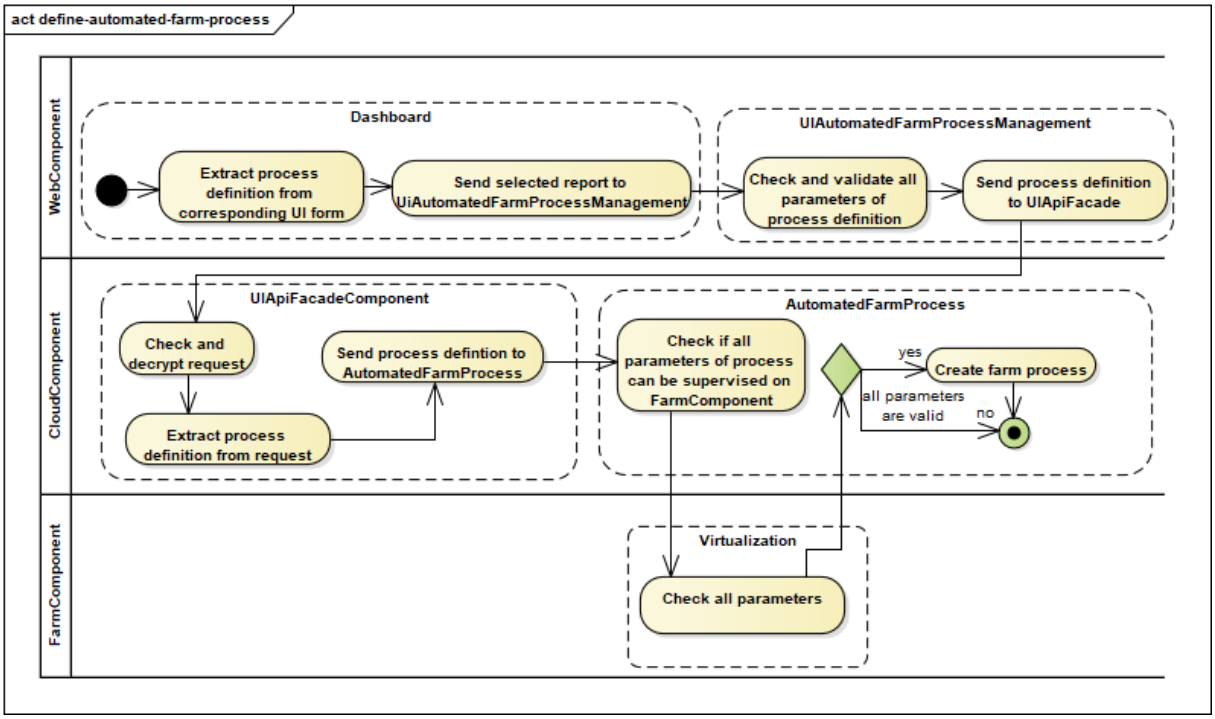

Figure 18: Activity diagram for defining an automated farm process

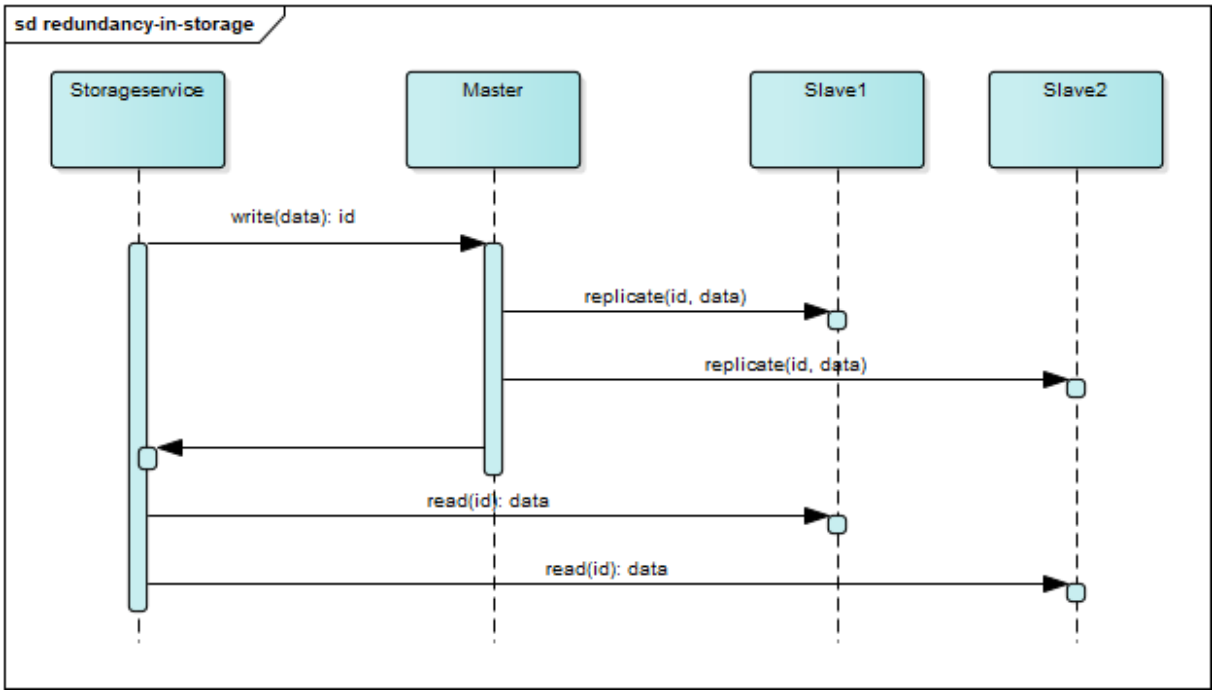

Figure 19: Sequence diagram for writing and reading data in redundancy of storage

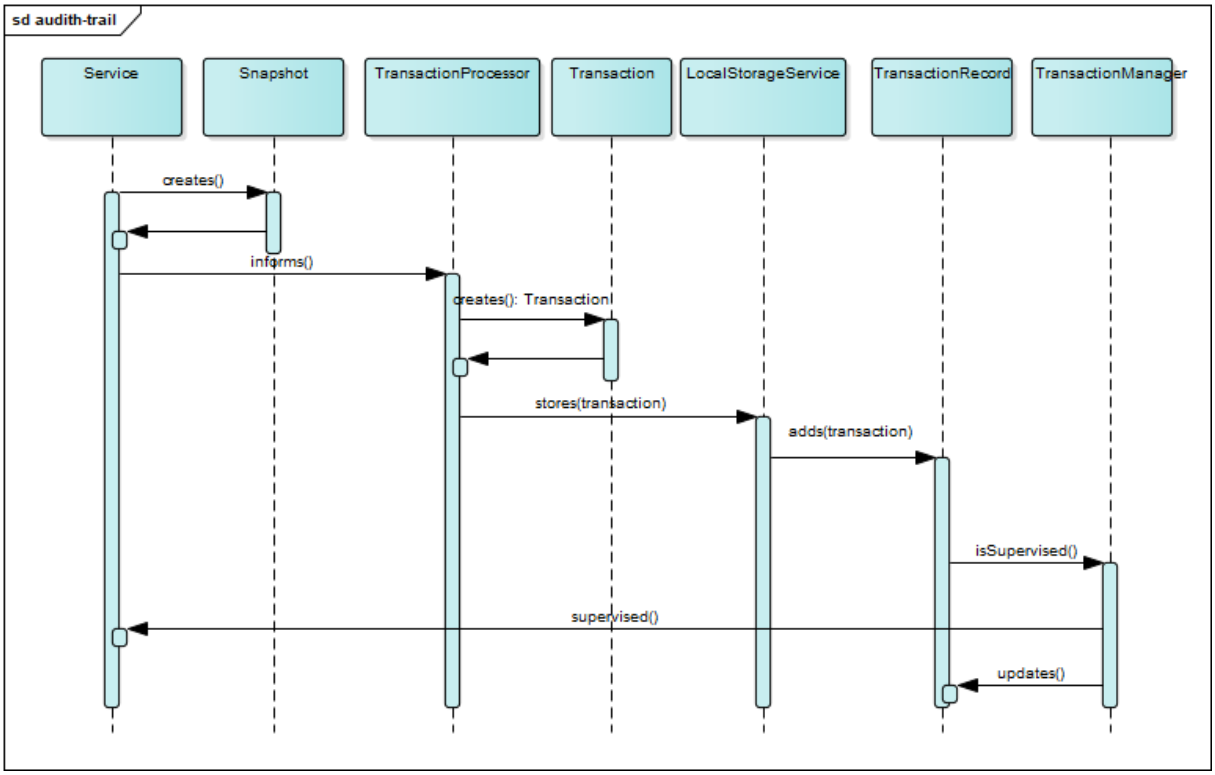

Figure 20: Sequence diagram for Audith-Trail pattern adaption for Transactions

8.2.4 Deployment view

Figure 21 gives an overview on how the different hardware devices are connected at the farm. Important is to mention, that there is always the same structure of devices: sensor -¿ bridge -¿ local server. This way devices can be scaled to fit the farm. In this case the bridge can not only delegate data flow to the local server but also pre-process data. In addition, since there are more devices on the farm that can run software. The audit trail pattern implementation for transactions can be distributed on several machines in order to ensure that no data is lost because of a failure during sending data with simply storing it locally. The RapsberryPi bridge can be connected directly to the kinect and thermal cameras. As long as there is no failure in the sensor or usb cable connection the transactions guarantee the arival of a measurement.

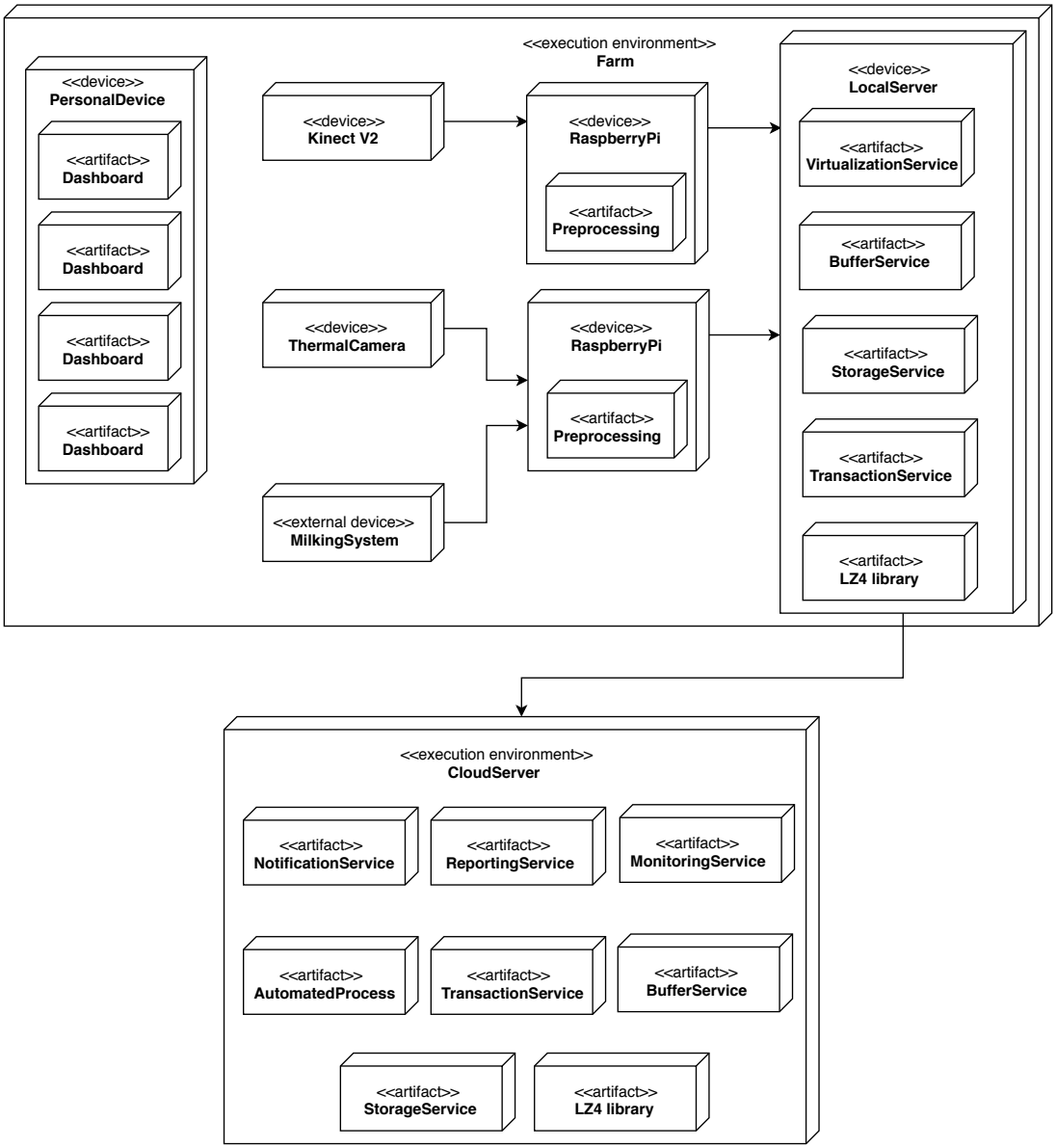

Figure 21: Deployment view for the Farmvolution system.

8.2.5 Use case view

The Use Case view (also known as the Scenarios view) captures Farmvolution's functionality as it is seen by the users. The use cases are specified in Section 4.5, as shown in Figure 4. The 6 use cases can be visualized in the Figures 22 - 27 below.

Use Case 01: Farmer defines automated farm process

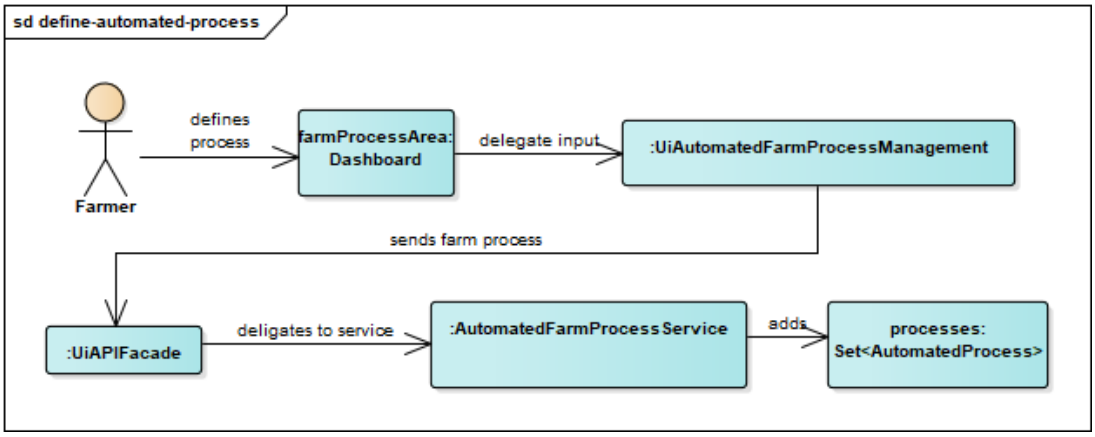

Figure 22: Use Case 01: Defining automated farm process

Use Case 02: Health Monitoring

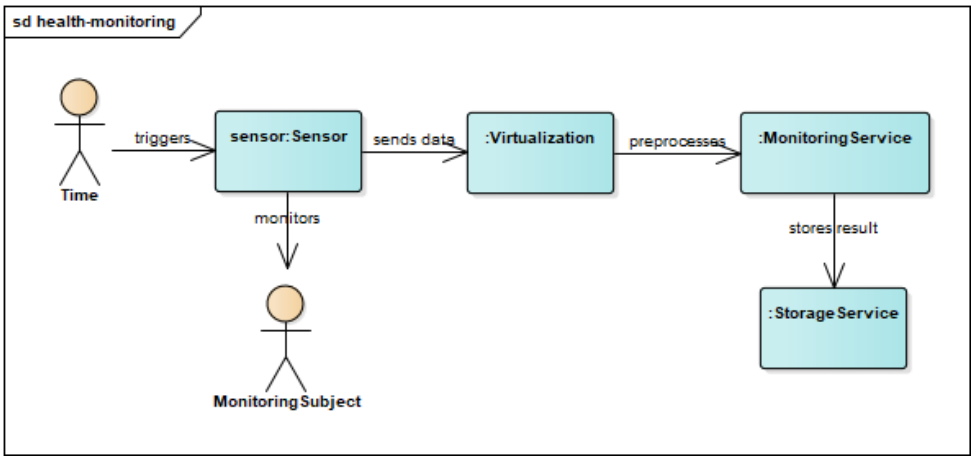

Figure 23: Use Case 02: Health Monitoring

Use Case 03: Supervising drinking and feeding behavior

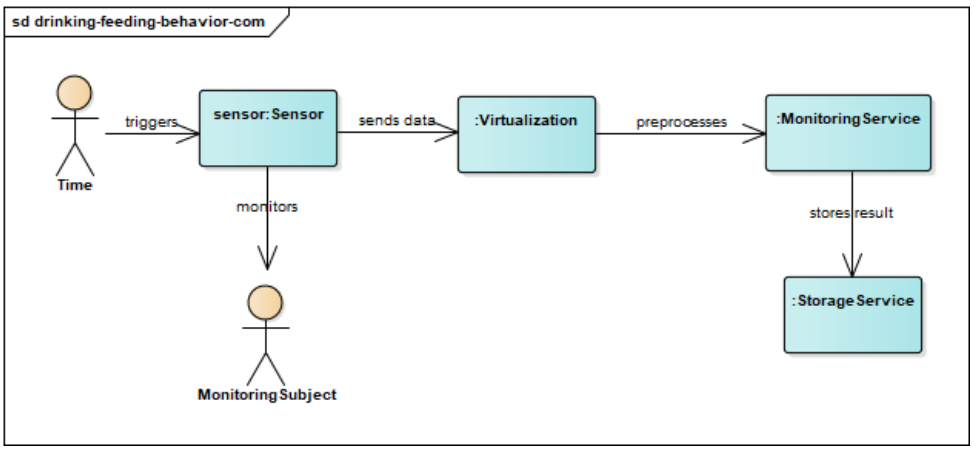

Figure 24: Use Case 03: Supervising drinking and feeding behaviour

Use Case 04: Reporting

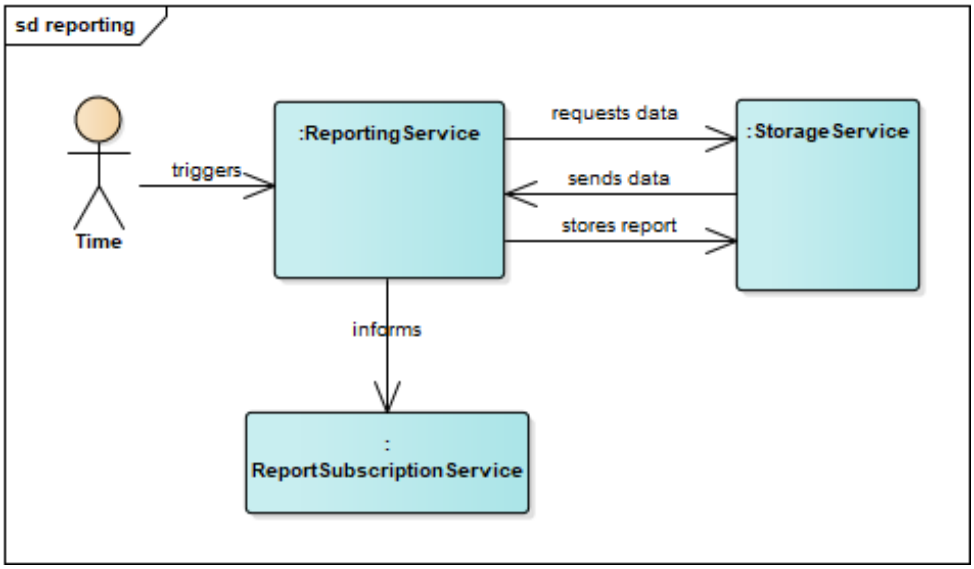

Figure 25: Use Case 04: Report Subscription

Use Case 05: Report subscription

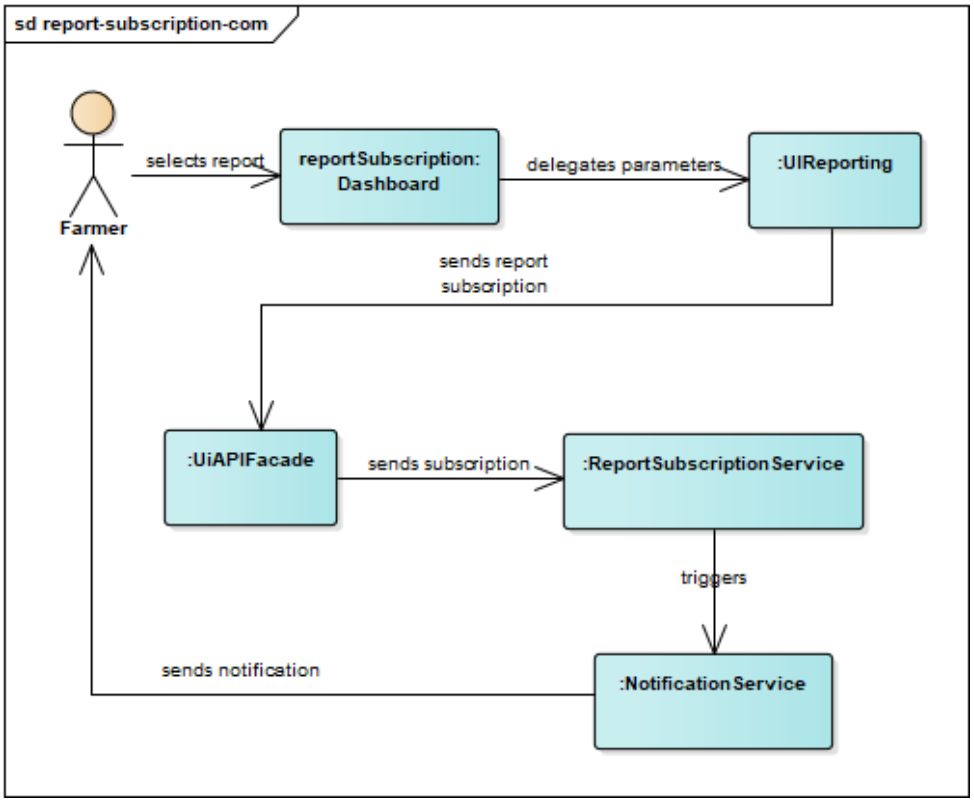

Figure 26: Use Case 05: Reporting

Use Case 06: Retrieve logistic demands

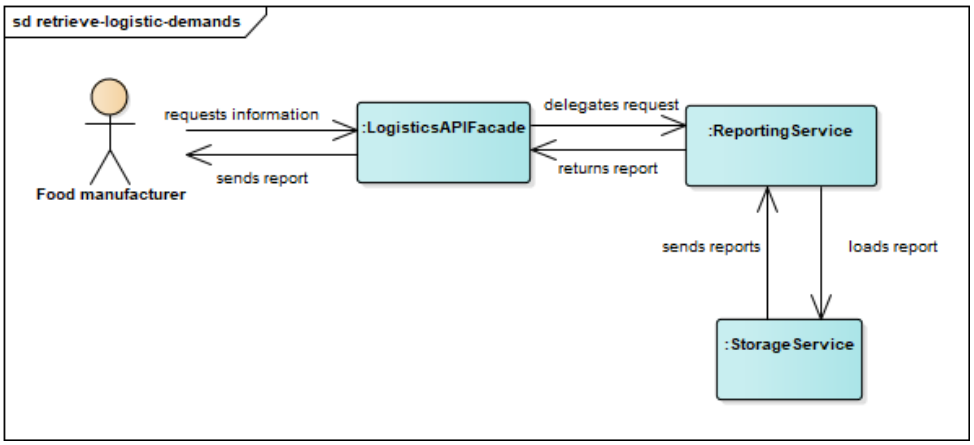

Figure 27: Use Case 06: Retrieve logistic demands

8.3 Decision views

Decisions of the software architecture for Farmvolution system are presented in this section. Initially, the decisions are included in Tables 35 - 41. Then, the decisions are included in Figures 28 and 29.

Name	**Database for storing farmer and food manufacturer information**
Decision	#1
Status	Approved
Problem/Issue	The chosen database has to be scalable, support structured data, and have quick CRUD operations.
Alternatives	*Couchbase Server*: Couchbase Server, originally known as Membase, is an open-source multi-model NoSQL document-oriented database software package and is high scalability. *MongoDB*: MongoDB is a highly scalable NoSQL database, that stores entries as JSON like objects(BSON).
Decision	The database solution chosen for storing Farmer and food manufacturer information is MongoDB
Arguments	MongoDB is an object-oriented, simple, dynamic, and scalable NoSQL database. It is based on the NoSQL document store model. The data objects are stored as separate documents inside a collection — instead of storing the data into the columns and rows of a traditional relational database. This will provide the ability to store multiple variable for each user. MongoDB is easy to implement and provides high performance, high availability, and automatic scaling. MongoDB is extremely simple to install and implement and is supported ya ll the major cloud providers.

Table 35: Software Design Decision: Database for storing farmer and food manufacturer information

Name	**Database for storing sensor data**
Decision	#2
Status	Approved
Problem/Issue	The chosen database has to be scalable, support time series structured data and have quick CRUD operations.
Alternatives	*Amazon DynamoDB*: Amazon DynamoDB is a fully managed proprietary NoSQL database service that supports key-value and document data structures and is offered by Amazon.com as part of the Amazon Web Services portfolio. DynamoDB exposes a similar data model to and derives its name from Dynamo, but has a different underlying implementation *CassandraDB*: Cassandra is a free and open-source, distributed, wide column store, NoSQL database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure.
Decision	The database solution chosen for storing Farmer and food manufacturer information is CassandraDB
Arguments	Cassandra is highly reliable, available and can handle massive time series data sets. Cassandra is fully tune able and let's you configure every aspect of the data replication. The number of replicas needs to be configured inside each datacenter and even enable replication across datacenter. In DynamoDB the number of replicas involved cannot be controlled as all the replications are performed automatically by AWS. You only provision the required throughput and DynamoDB makes sure you have enough partitions to handle the load. This becomes difficult to manage when sensor data is used.

Table 36: Software Design Decision: Database for storing sensor information

Name	**Cloud Platform**
Decision	#3
Status	Approved
Problem/Issue	The chosen cloud platform has to support the software components of the system, database, the machine learning algorithm and generation ofreports. It has to be scalable, reliable and should have high performance.
Alternatives	*Amazon Web Services(AWS)*: It is one of the most popular cloud platforms and is widely used in the market *Google Cloud*: Google Cloud Platform, offered by Google, is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products *Microsoft Azure*: Microsoft Azure is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through a global network of Microsoft-managed data centers
Decision	The cloud platform chosen to store the components is Amazon Web Services.
Arguments	Amazon web services have high reliability, scalablility and performance. It also provides services for ensuring elasticity and fault tolerance through load balancing, component replication and also provides virtual environment load software and services.

Table 37: Software Design Decision: Cloud Platform

Name	**Machine learning Algorithm for detection of Mastitis**
Decision	#4
Status	Approved
Problem/Issue	The chosen machine learning algorithm should provide high accuracy for detection of mastitis, be saclable and process large amounts of thermal images.
Alternatives	Artificial neural networks or connectionist systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains.
Decision	The Machine learning algorithm which will be used for detection of Mastitis will be Self Organising Map (SOM) networks
Arguments	Based on our literature review about detection of Mastitis in cattle we have found a paper which produces 96% accuracy for detection using thermal images [9]. The algorithm can be easily deployed on the cloud service and can be scaled in order to generate predictions in the required time frame

Table 38: Software Design Decision: Machine learning algorithm for detection of Mastitis

Name	**Machine learning algorithm for detection of respiratory disease for pigs**
Decision	#5
Status	Approved
Problem/Issue	The chosen machine learning algorithm should provide high accuracy for detection of pig coughs, be saclable and process large amounts of audio and image data from Kinect V2.
Alternatives	*Hidden Markov Model(HMM)*: A hidden Markov model is a statistical model in which the system to be modeled is a Markov process with unknown parameters. Gaussian Mixture Model (GMM) A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. *Support Vector Data Description (SVDD):* Support Vector Data Description is a variant of the support vector machine, which has high generalization performance by acquiring a maximal margin in one-class classification problems.
Decision	The machine learning algorithm which will be used for detection of respiratory disease for pigs is Support Vector Data Description (SVDD) as an early anomaly detector and a respiratory disease classifier, respectively
Arguments	Based on our literature review about detection of respiratory diseases in pigs we have found a paper which produces 94% accuracy for detection using audio data [2]. The algorithm can be easily deployed on the cloud service and can be scaled in order to generate predictions in the required time frame

Table 39: Software Design Decision:Machine learning algorithm for detection of pig coughs

Name	**Development Language for Machine learning algorithms**
Decision	#6
Status	Approved
Problem/Issue	The chosen development language will be used for implementation of the machine learning algorithms in the farmvolution system which should be well documented, compatible with the cloud platform and should be widely used.
Alternatives	*Java*: Java is a general-purpose computer-programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. *Python*: Python is an interpreted high-level programming language for general-purpose programming
Decision	The language used for development of machine learning algorithms will be Python
Arguments	Python consists of machine learning libraries which can be used for building both the algorithm for cattle and pigs. It is also an well documented language for processing images and audio signals and can be easily deployed on any cloud service.

Table 40: Software Design Decision:Development language for Machine learning algorithms

Name	**Development Language for Web Interface**
Decision	#7
Status	Pending
Problem/Issue	The chosen development language will be used for implementation of the Web Interface in the Farmvolution system which should be well documented, compatible with the cloud platform and should be widely used.
Alternatives	*JavaScript*: JavaScript is a commonly used scripting language for making web pages interactive and developing web applications. *Python*: Python is an interpreted high-level programming language for general-purpose programming
Decision	The language chosen for the web interface is javascript
Arguments	Java script is an easy to use web interface language which contains a variety of functionalities for development of the of a web application

Table 41: Software Design Decision:Development language for Web Interface

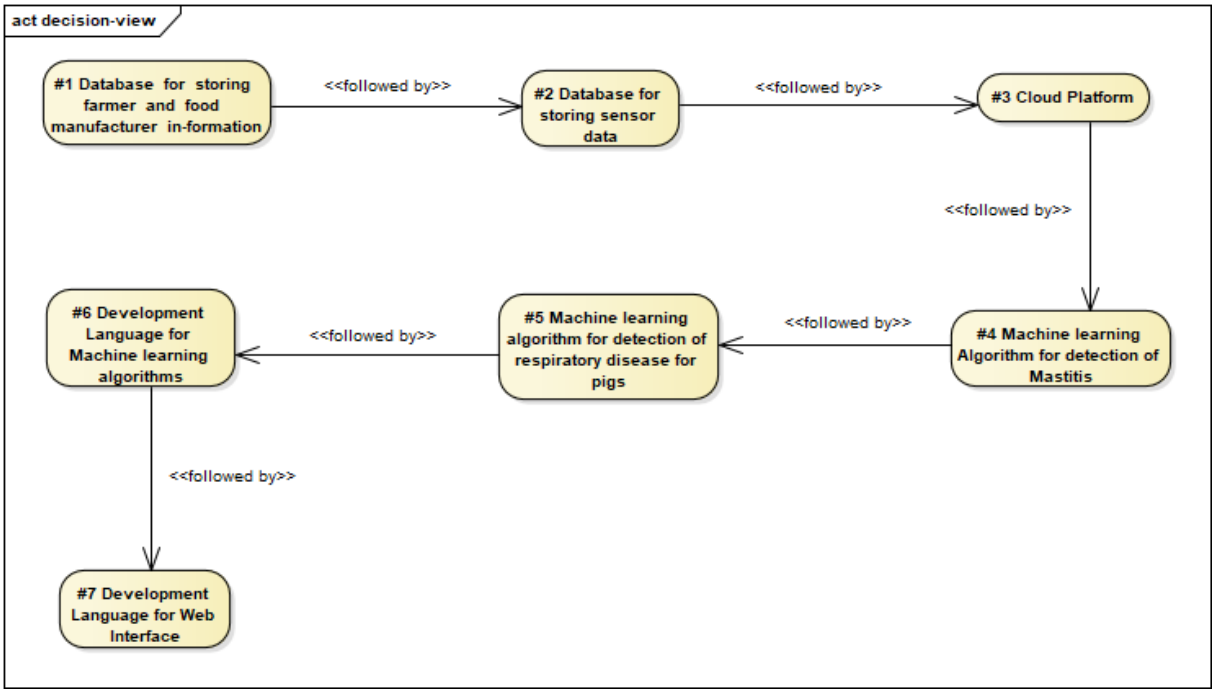

Figure 28: Chronological overview on decisions

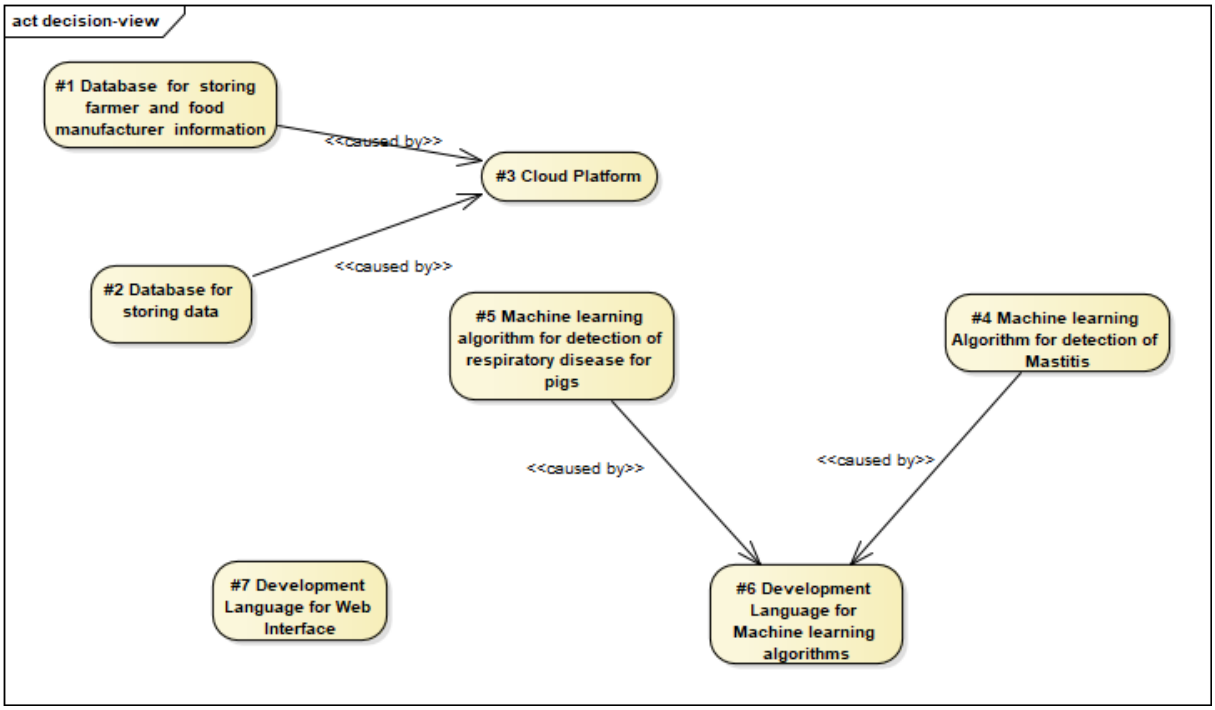

Figure 29: Overview on decisions and their relationship to other decisions

9 Architecture Evaluation

This chapter focuses on evaluating the architecture created in the previous chapters. Firstly, the requirements are checked to make sure that all the important requirements are fulfilled. Then, the Architecture Trade-Off Analysis Method is used to evaluate the architecture. Finally, conclusions of the architecture are listed, as well as possible improvements.

9.1 Requirements Verification

The requirements for the system are decided based on the Stakeholders of the Famvolution system described in Section 4.2 (Stakeholders). Baased on stakeholder's interests, 3 main key drivers (Section 4.3) and 5 high level requirements (Section 4.4) were created. These represented the basis for the system's functional requirements (Section 4.6) and non-functional requirements (Section 4.7).

9.2 Architecture Trade-off Analysis Method

In this section the steps of Architecture Trade-Off Analysis Method (ATAM) are discussed. Also, there is a discussion on how this method is used during the elaboration of the architecture.

9.2.1 Evaluation Steps

ATAM is a method for evaluating software architectures relative to quality attribute goals. ATAM evaluations finds architectural risks that might restrain the achievement of an organization's business goals. Table 42 shows the steps taken for the ATAM.

Step	Description	Initiator	Documented in
1	Present ATAM	Moderator	Section 9.2.1
2	Present business drivers	Business stakeholder	Section 3
3	Present the architecture	Architect	Section 4.1, 6, 7, 8
4	Identify architectural approaches	Architect	Section 9.2.2
5	Generate quality attribute utility tree	All	Section 9.2.3
6	Analyze architectural approaches	Architect	Section 9.2.3, 9.2.4
7	Brainstorm and prioritize scenarios	All	Section 9.2.4
8	Analyze architectural approaches	Architect	Section 9.2.4
9	Present results	Moderator	All Sections

Table 42: Steps of the ATAM and their elaboration in this document

Moderator
Moderator is the person whose responsibilities are leading the ATAM meetings, discussions and brainstorm sessions.

Architect
Architect is a person or a group of persons that are responsible for designing proposals and elaborate models for the architecture.

9.2.2 Architectural Approaches

This subsection describes the main architectural approaches used for the creation of the Farmvolution system. These approaches were briefly discussed in Sections 6 to 8 as well.

The first and most important decision is the separation of the system into 3 main components: the farm component, the cloud component and the web UI, as presented in Section 8 and shown in Figure 8. This method allows for a scalable and high performance solution. This method is easily maintainable and cost efficient.

Following is a list of decisions organized on the main components.

Farm Component

1. Virtualization - data coming from different sensors can be wrapped into a general form to ease the computations. Virtualization also provides a single point of access to the data by aggregating it from a wide range of data sources.

2. Bridges - sensors first send data to the bridges, which act as intermediaries between them and the local server. At the bridge level, data is pre-processed before it is send to the server. The pre-processing step is done in order to sample and discard redundant data in order to perform faster computations without affecting the quality of the output.

3. Local server - The local server receives pre-processed data from the bridges, compresses it, then sends it to the cloud component to analyze it. As a redundancy mechanism, it also allows data to be stored for up to 24 hours if teh cloud becomes unavailable.

4. Audit Trail Pattern - this transactional pattern ensures the correct process and safe transfer of data from sensors to the local server. This pattern improves the redundancy of the system, making it more reliable.

Cloud Component

1. Actual use of cloud component - making use of a cloud component to run the machine learning algorithms used to analyze the data received from the server is much more efficient than deploying high-end severs, as this method is highly scalable and easily maintainable.

2. Buffering - The risk of losing connection to the Farm Component is mitigated using buffers at the cloud level as well. Heart-beats are sent to check the connection status. When the connection is re-established, the buffered data can then be streamed to the local server from the farm component.

3. Redundant storage system - the cloud component also stores the data for later access. Clearly, any failure in storage nodes could result in data loss. This risk is mitigated by the use of a replica set, containing a master database which is replicated by 2 slave databases.

Web UI Component

Modular structure - using modern Single-Page Application web interfaces. The UI component is easier to maintain and more user-friendly.

9.2.3 Utility Tree and Scenarios

In this section the quality attribute utility tree of the architecture is provided.

Table 43 are given an indication of importance versus cost to the entire system deployment. On the left side of the scenario is the Importance and the Cost attribute while on the right side of the scenario is the total score.

	Cost		
Importance	**High(1)**	**Medium(2)**	**Low(3)**
High(3)	3	6	9
Medium(2)	2	4	6
Low(1)	1	2	3

Table 43: Scenarios prioritization rankings

Cost = Implementation effort
Importance = Contribution to overall success

The main key drivers (availability, performance and reliability) are analyzed in Table 44. These key driver were defined in Section 4.3.

Quality Attribute	**Requirement**	**Score**	**Scenario**
Reliability	Waterfall dependencies to be avoided	9 (H,L)	Table: 45
	Minimize the duration of downtime	6 (H,M)	Table: 46
	Always one database instance available	3 (H,H)	Table: 47
Performance	Fast detection of rule violation	6 (M,L)	
	Fast notification delivery	9 (H,L)	
	High throughput over video and audio streaming	3 (H,H)	
Reliability	Safe hardware for the animals health	3 (H,H)	
	Storage must be scalable	2 (M,H)	
	24 hours data storage for each sensor	2 (M,H)	
	Data must be reserved during sensor failure	6 (H,M)	

Table 44: Quality attribute utility tree

9.2.4 Analysis of Scenarios

This subsection contains some scenarios described in the previous section. They are presented in the tables below and are followed by descriptions of the sensitivities, tradeoffs, remaining risks and non-risks.

Scenario 1	One Component fails during collection, sending or processing data				
Q-Attribute	Reliability (Table 18)				
Environment	Data of a certain sensor is collected				
Stimulus	One part of Farm- or CloudComponent fails				
Response	99 % Reliability				
Decisions	**Requ.**	**Sensitiv.**	**Tradeoff**	**Risk**	**Non-Risk**
Transaction Audit Trail	NFR1-3	S1	TO1	R1	
Buffering	NFR1-3				NR2
Redundancy of Storage	NFR1-3				NR3
Reasoning	The decisions reduce the loss of the data and will nearly always rely on correct data				
Arch. model	Audit Trail Pattern for transactions, Buffering, Redundancy of Storage components				

Table 45: ATAM Evaluation of Scenario 1

Sensitivities

S1 = No transaction will be created if sensor fails after collecting and transmitting data to bridge and hence no backup is available.

Trade-offs

TO1 = Several retries if transaction process of certain sensor process wasn't triggered.

Remaining Risks

R1 = No reliable data will be stored if collection of data failed. Thus, NFR1-3 might not be fulfilled in this case.

Non-Risks

NR2 = System will not fail with creating reliable data even if Farm/CloudComponent cannot communicate. Thus, NFR1-3 will be fulfilled in this case.

NR3 = CloudComponent is able to read and write reliable data from cloud storage even if one database is not available. Thus, NFR1-3 will be fulfilled in this case.

Scenario 2	Farm Component has no internet connection				
Q-Attribute	Availability (Table 16)				
Environment	Normal operation				
Stimulus	Local internet connection of the farm is not abvailable.				
Response	minimum 50% of all components are available				
Decisions	**Requ.**	**Sensitiv.**	**Tradeoff**	**Risk**	**Non-Risk**
Buffering	NFR0-3				NR4
Redundancy of database instances	NFR0-5				NR5
Reasoning	Those decisions will ensure availability of certain parts of the system				
Arch. model	Buffering, Redundancy of database instances				

Table 46: ATAM Evaluation of Scenario 2

Non-Risks

NR4 = Neither FarmComponent nor CloudComponent fail due to sending data back and forth.

Thus, NFR0-3 is fulfilled in this case.

NR5 = Redundancy of database instances ensures availability of database.

Thus, NFR0-5 is fulfilled in this case.

Scenario 3	Unavailable internet connection causes high amount of data for upload from the local server to the cloud component				
Q-Attribute	Performance (Table:17)				
Environment	Components of System cannot communicate				
Stimulus	The internet connection was lost to the cloud or the local hard drive failed				
Response	Video and audio data size has to be reduced by ∼90%				
Decisions	**Requ.**	**Sensitiv.**	**Tradeoff**	**Risk**	**Non-Risk**
Store data locally in case cloud failed	NFR1-1				NR6
Pre-process data locally	NFR1-1				NR6
Reasoning	Even for big data that needs to be uploaded, the system has adequate upload performance regarding the amount of data				
Arch. model	Store data locally in case of no internet connection, pre-process data locally				

Table 47: ATAM Evaluation of Scenario 3

Non-Risks

NR6 = The pre-processing reduces the data and in case of a longer period of storage the upload of stored data is manageable.

Thus, NFR1-1 is fulfilled in this case.

10 System Evolution

The current version (version 1.0.0 - November 2018) of the Farmvolution can be efficiently used in livestock farms. The previous chapters proposed solution for the architecture of the farmvoltuion system. The system provides all the required functionality and has good features, however they system needs to keep evolving in order to thrive in the competitive and evolving smart-farm market. To achieve this the system needs to add new features and improve the existing features which are discussed in the following section.

10.1 Mobile Application

An mobile application will provide a great asset in communicating with the customers such as Food manufacturers, Farm owners and Farm workers. The mobile application will be developed for Android, iPhone and Windows Phone. Users can perform all actions which are possible using the web portal using the mobile application. It will provide farm workers with notifications when disease has been detected and and information overview to the farm owners about the diseases detected and livestock information. The mobile application will be published on Google Play, the Apple App Store and the Microsoft App Store and will be made available for the customers for free.

10.2 Localization

The farmvolution system plans to launch the system across Europe in 2021 Q1 as mentioned in 3.7. Localization an app and integrating a feature to choose a language should not be difficult to provide.

10.3 Expansion in crop domain

The farmvolution system will expand into the crop domain in 2022 Q1 as mentioned in 3.7 and subsequently in Europe in 2023 Q1. The main focus of this expansion is to provide crop farmers with features such as health monitoring and disease detection in farms. This version of system will follow the same infrastructure however based on the crops and the features required by the farmer the farm component and cloud component will modified accordingly using the appropriate sensors and algorithms respectively. Integrating a new machine learning model for health monitoring and disease detection for crops and the pre-processing of new data need to be processed separately and scaled which is feasible with the current infrastructure.

10.4 Farmvolution system versions

Table 48 shows the versions of farmvolution system

Version	Description
1.0	Current version
1.1	Localization of languages
1.2	System integration with a mobile application
2.0	System integration into the crop domain

Table 48: Famrvolution system versions

11 Acknowledgements

We would like to express our appreciation to our coach, Dan Chirtoaca, for reviewing our iterations and providing feedback on our drafts of this document, and to Paris Avgeriou for his lectures on Software Architecture.

We thank the students of Team 3 (A.Lukjanenkovs, M.Abd Elrehim, G.Knap, and K.Gkikopouli) for providing feedback and recommendations on version 0.3.0 of this document.

We thank the students of Team 2 (Ruben Kip, Ana Roman and Orest Divintari) for providing feedback and recommendations on version 0.7.0 of this document.

12 Glossary

ADD - Attribute-Driven Design

API - Application Programming Interface

ASD - Architectural Significant Driver

ATAM - Architecture Trade-Off Analysis Method

AWS - Amazon Web Services

CNFR - Commercial Non-Functional Requirement

COSA - Component Oriented Software Architecture

CPU - Central Processing Unit

DB - Database

DT - Development Team

FLIR - Forward Looking Infrared Radiometer

FO - Farm Owner

FR - Functional Requirement

FW - Farm Worker

GUI - Graphical User Interface

HLR - High-Level Requirements

HTTP(S) - Hypertext Transfer Protocol (Secure)

MT - Maintenance Team

MTTF - Mean Time To Failure

NFR - Non-Functional Requirement

PO - Product Owner

QA - Quality Attribute

REST - Representational State Transfer

RFID - Radio-Frequency IDentification

ROI - Return Of Investment

TCO - Total Cost of Ownership

TNFR - Technical Non-Functional Requirement

TTM - Time To Market

Bibliography

[1] Len Bass, Mark Klein, and Felix Bachmann. "Quality Attribute Design Primitives and the Attribute Driven Design Method". In: *Software Product-Family Engineering*. Ed. by Frank van der Linden. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 169–186. ISBN: 978-3-540-47833-1.

[2] Yongwha Chung et al. "Automatic detection and recognition of pig wasting diseases using sound data in audio surveillance systems". In: *Sensors* 13.10 (2013), pp. 12929–12942.

[3] European Commission. *How many people work in agriculture in the European Union? An answer based on Eurostat data sources*. 2013. URL: https://ec.europa.eu/agriculture/sites/agriculture/files/rural-area-economics/briefs/pdf/08_en.pdf.

[4] Michael French. "Thermal imaging technology: What can it do for the dairy industry?" In: (September 2016). URL: https://www.dairyreporter.com/Article/2016/09/02/Thermal-imaging-technology-What-can-it-do-for-the-dairy-industry.

[5] M. Grand. "Transaction Patterns - A Collection of Four Transaction Related Patterns". In: (1999). URL: https://hillside.net/plop/plop99/proceedings/grand/plop_99_transaction_patterns.pdf.

[6] P. B. Kruchten. "The 4+1 View Model of architecture". In: *IEEE Software* 12.6 (November 1995), pp. 42–50. ISSN: 0740-7459. DOI: 10.1109/52.469759.

[7] *LZ4 high speed compression benchmark*. URL: https://lz4.github.io/lz4/.

[8] Mittek M. et al. *Health Monitoring of Group-Housed Pigs using Depth-Enabled Multi-Object Tracking*. 2018. URL: http://homepages.inf.ed.ac.uk/rbf/VAIB16PAPERS/vaibmittek.pdf.

[9] Sandhya Samarasinghe, Manishi Kohli, and Don Kulasiri. *Neural Networks for Robotic Detection of Mastitis in Dairy Cows: Netherlands and New Zealand Perspectives*. September 2018.

A Appendix

A.1 Full Concern Matrix

	Weight (%)	**Timeliness**	**Performance**	**Reliability**	**Availability**	**Usability**	**Modifiability**	**Safety**	**Security**	**Portability**	**Reusability**	**Integrability**	**Interoperability**	**Testability**
FO	100.00	5	10	10	10	8	4	7	7	7	0	6	2	0
PO	100.00	7	10	9	9	8	7	4	7	7	2	3	2	2
FW	90.00	8	7	7	9	10	5	8	8	3	0	1	0	0
FM	75.00	10	9	10	9	8	2	7	5	1	0	0	2	0
MT	65.00	7	7	8	9	5	9	4	7	7	8	6	8	8
DT	60.00	3	8	8	6	7	7	4	7	7	8	8	7	8
Total		34.85	**42.4**	**42.2**	**43.3**	38.45	27.05	32.45	33.7	26.2	12	18.6	14.9	12

Table 49: Concern Matrix

A.2 Full Risks Tables

Label	**Risk**	**Likelihood**	**Severity**	**Responsibility**	**Trigger**	**Consequences**	**Prevention**	**Reaction**
TR1	Failed internet connection	Medium	Medium	Farm Owner	Loss of Heartbeat signal from FarmComponent	Reduced quantity and quality of the reported information	-	Buffer sensor data locally. Farmvolution will notify farm owner and maintenance team about the malfunction.
TR2	Incorrect disease diagnosis	Low	Medium	Architect	Incorrect data	Spreading disease, reduced production	-	-

Table 50: Technical risks and preventive measures taken to mitigate these risks.

Label	Risk	Likelihood	Severity	Responsibility	Trigger	Consequences	Prevention	Reaction
BR1	Wrong budgeting plans	Medium	Medium	Product Owner	inability to deliver on time	Delay of initial release	Allocate emergency resources	Reduce project scope
BR2	Too much competition	Medium	Medium	-	Reduction in sales due to competition	Reduced profitability	Gather information on emerging competitors	Perform surveys on customers to find new possible features.
BR3	Project scope is too large	Medium	High	Architect	inability to meet milestones	late release of major versions and components	Determine realistic scope during architecting phase	Stop development of least important features

Table 51: Business risks and preventive measures taken to mitigate these risks.

Label	Risk	Likelihood	Severity	Responsibility	Trigger	Consequences	Prevention	Reaction
OR1	Communication difficulties within team	Low	Medium	Product Owner	Developer indicates to struggle with team dynamics	Productivity of development team is low	Have regular stand up meetings and team building events	Team members with problems should resolve the issue with HR.
OR2	Bad requirements engineering	Low	High	Product Owner	Product Owner does not think requirements are being met	System behaviour does not align with product vision	Frequently include feedback from product owner when eliciting requirements	Reiterate over requirements until they match the vision of the Product Owner
OR3	Lack of expertise within team	Low	High	Product Owner	Developers/-Maintainers are unable to deal with a problem	Issue cannot be resolved on time	Ensure that employees with a wide variety of skills are hired	Hire someone with the required expertise as soon as possible

Table 52: Operational risks and preventive measures taken to mitigate these risks.

Label	Risk	Likelihood	Severity	Responsibility	Trigger	Consequences	Prevention	Reaction
IR1	GUI is hard to use	Low	Low	Development team	-	Reduced number of clients	Redesigning the GUI	More money need to be spend on development team
IR2	New libraries are hard to implement	Low	Medium	Development teams	development team suspects delays	Feature might be delayed	Pay close attention to integrability reviews when choosing libraries	Investigate alternatives
IR3	New hardware components are incompatible	Low	Medium	Maintenance team	A new sensor does not operate as expected	System must be adapted before the component can be used	Test whether a new component is compatible	Create a virtualization module for the component

Table 53: Implementation risks and preventive measures taken to mitigate these risks.

Label	Risk	Likelihood	Severity	Responsibility	Trigger	Consequences	Prevention	Reaction
Other1	Government increases restrictions on animal farming	Very Low	High	Law Department		Increased regulation on animal farming can lead to a shrinking market for our product	Expand into the crop market	Shift resources from animal related products to products that monitor crops

Table 54: Other risks and preventive measures taken to mitigate these risks.