ETSN15,

Requirements Engineering

Vultus - Anomaly Detection Software Requirement Specification

Version 1.1

Edvin Havic, Martin Johansson, Niklas Bruce, Oscar Rydh, Peter Skopal

supervised by

Daniel Helgesson @ LTH

November 27, 2017

Contents

1 Change Log

Table 1: Change Log

Date	Version	Changes
2017-11-20	R1	Baseline
2017-11-21	R1.1	Added Use Cases Added Tasks Added Functional Requirements for each Task Added Quality Requirements Added Sequence Diagram

2 Introduction

The goal of Vultus is to provide the farmers with tools to help increase their efficiency and increase yield. This is done by analyzing satellite data of their farmlands, detecting possible problems and notifying the user. This means that the farmer can take preventive action and thereby increase their profit. An example of farming problems is that 60% of all nitrogen used in farming is wasted [4]. Eliminating this will not only give greater profits for farmers but only give environmental benefits, as over-fertilization is one of the greatest environmental problems today[4].

Vultus has previous experience in plant life analysis. Their existing systems provides the farmers with information and instructions to maximize the yield of their fields. Now it is time to extend this system. The goal with this requirement document is to specify a tool for anomaly detection in the users farmlands. The idea is that instead of using pesticides evenly across large fields, give the user the ability to use them in areas where the system has detected an anomaly.

3 About the Document

3.1 Purpose of Requirements Specification

This Software Requirement Specification(SRS) was created to set requirements on the anomaly detector developed by Vultus. The goal of the document is not to provide specifications about the implementation of the anomaly detection, but to specify the connection between differ

Vultus has approached us to help them with the requirements for their anomaly detector. The goal with the requirements is not to provide an architecture for the system, but rather provide a document that is well defined about how the system should work. This means, for example, that a software developer should be able to read this document and then be confident in how he should create the system. The requirements themselves are defined on different levels providing information on different levels of the domain.

3.2 Scope of Requirements Specification

This project has multiple goals, definitions and scope depending on the stakeholder. The following sections will describe it more in detail. The main goal of the anomaly detection is to provide the customers of Vultus with even more value in form of an early detection system for pests.

4 Definitions and Terms

The following definitions will be used throughout the document.

Farmer Management System (FMS) is an management system used by farmers, keeping track of many different aspects on a farm. Examples are keeping track of crops, seeds, revenue, growth etc.

Copernicus is an EU project aiming at retrieving and developing an information service based on satellite data.

NASA is a well known space organisation in the US, which as well as Copernicus offers satellite data.

User is referred to any person using or being affected by the FMS, e.g. when a farmer delegates a task from the FMS to a worker on a farm.

Vultus Engine is an already existing system within Vultus. It processes satellite images making decisions on crop health. It will act as an part communication with the system specified in this SRS.

Yield is the output of a field and how much product this delivers over one growing cycle.

5 The System in Context

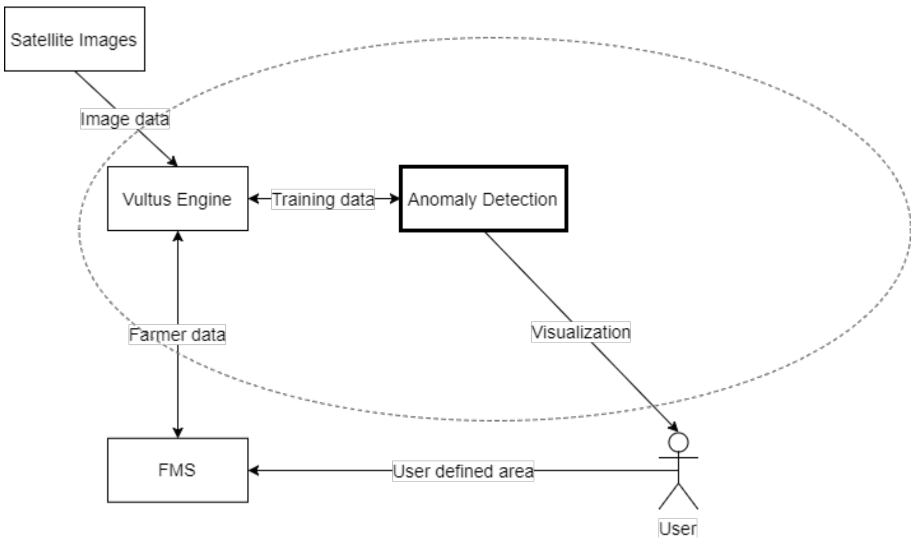

Figure 1: Context diagram of the entire domain, our system is the anomaly detection

Weeds, insects, fungus, drought and deficiencies are among the problems plaguing farmers worldwide. They cause tremendous damage to crops, and are often combated and controlled. The standard method to monitor these damages is by walking around the field. Walking limits the frequency that the field can be checked, and therefore issues go undetected for longer periods of time and cause tremendous damage.

The proposed product idea is to apply machine learning to remotely sensed satellite data to detect and predict crop damage. This early detection system would at the earliest possible time detect problems in the field, and possibly even predict upcoming damage. This would warn the farmer and allow on the ground inspection and appropriate action to be taken. The goal is not to diagnose the problem, as this is almost impossible to do from remotely sensed data, however, using machine learning, it should be possible to notice indicators of upcoming crop damage to predict when areas will be damaged.

The satellite data would be multi-spectral, and a direct indication of vegetation status. Common such indices are the Normalised Difference Vegetation Index (NDVI), that compares red and near infrared light to accurately depict crop status. When specific areas of a field have a decrease of those index values, it is likely due to crop damage. The proposed system would look at a time series of satellite data, taken every 5 days, of specific fields. This time series exists from previous years, dating back to the 80s. The system should be able to detect when certain areas of fields grow worse, automatically. However, parts of the field that are underperforming each year should 2 not be detected as anomalies. These anomalies would be sent to the farmer to warn about current damages. The system will communicate with already existing modules, both with existing backend, but also with an interface as a separate module in a farmer management system [3].

The satellite data used in the product is generated by a different organizations, in the case of data in regard to Europe the data comes from the Copernicus project. Which is a project financed by the European Commission and the observation infrastructure is provided by ESA(European Space Agency). Similar data can be retrieved by similar organizations like NASA. this information would then provide Vultus with the information needed to predict anomalies and calculate crop health. The system referred to in this SRS is depicted as a context diagram in figure 1.

5.1 Stakeholders

We have identified multiple stakeholders affected by the project. Here we will describe each stakeholder and their relation to the system.

5.1.1 Customer

Given that the analysis can be generalized across crops and climate zones, every farmer could be a potential customer. Even smallholder farmers without internet can get SMS notifications. We believe this kind of analysis could drastically reduce crop loss. Pests alone reduce global yields by over 30%. Most likely the initial customers would be industrialized farms with larger areas due to increased purchasing power, decreased ability to scout by foot and technological maturity.

5.1.2 FMS

Farm Management Systems (FMS) make the lives of farmers easier, and providing a wider set of services for the customer is beneficial for the FMS since this benefits the end user, i.e. the farmers. In this case Vultus will use their product as an integration module for their existing system. The FMS have interest in the system to make their system more desirable and profitable.

5.1.3 Copernicus

While Copernicus is not directly involved in the project, it's in their best interests that their product is used. Vultus uses Copernicus to provide satellite images over the farms that they can analyze to provide their service.

5.1.4 Investors and Product Owner(Vultus)

While development cost is an initial risk for both investors and Vultus, extending the functionality of the systems offered by Vultus makes them more competitive on the market by having a wider range of products.

5.2 Amazon Web Services

5.2.1 Other stakeholders

Crop insurers have a huge interest in reducing crop loss due to expensive payouts (often hundreds of thousands of kr). Food security is a large issue for many developing countries. Early warnings can provide the chance to import food when food production is hurt, as well as decrease the loss from pests. Furthermore, all governments are concerned with guaranteeing domestic food security, hence subsidies on food production. The product would further their work in guaranteeing food security.

5.3 Objectives

Vultus objectives of the system are:

- to develop a system being able to detect crop anomalies
- to integrate the system into already existing parts of the organisation
- to increase overall crop yield

6 System Requirements

6.1 System Goal

The goal of this system is to increase quality and functionality to the already existing product, by being able to detect anomalies in the growth cycle of crops. More specifically: *The system should provide a reliable early detection system of anomalies in crops growth cycle.*

6.2 Data Requirements

The system uses different types of data to analyze and detect anomalies in the fields. For the system to work correctly the data must both be of sufficient quality and quantity. The system uses machine learning to detect anomalies and for this detection to work we need sufficient, specific and quality data.

6.2.1 Farm Data

The system must have data about the farm it is to analyze.

- An id keeping track of the current farmer
- Field data
  - Field coordinates indicating where on the globe the crop field is located.
    - GeoCoordinates following rfc standard [5]
  - Crop data
    - Crop type
    - Planting dates
    - Fertilizer type
    - Fertilizing dates
    - Special events during crop growth

6.2.2 Analyzed Data

This data is crucial for the system actually helping in the decision making on anomaly detection.

1. A crop health model containing differences based on a combination of the satellite imagery and field data.
2. A severity classification based on the level of anomality found in a field area
3. The area owned by the user to be used as a basis for the analysis.

6.2.3 Data Dictionary

- **GeoCoordinates** is a RFC standard defining how a geographic coordinate is described using the format of a JSON.
  Example
  ```json
  {
    "type": "Feature",
    "geometry": {
      "type": "Point",
      "coordinates": [55.712449, 13.214965]
    },
    "properties": {
      "name": "Vultus AB"
    }
  }
  ```
- **Crop type** is an identifier indicating what crop are growing on a certain field.
  Example
  1. corn
  2. potatoes
  3. tomatoes
  4. etc.
- **Planting dates** are identifiers indicating when a certain crop in a field has been planted
  Example
  2017-04-14T12:00:00+00:00
- **Fertilizing dates** are identifiers indicating when a certain crop in a field has been fertilized.
  Example
  2017-04-16T14:00:00+00:00

6.3 Functional Requirements

In this section we will define the functional requirements of the system. The system consist of many different parts that communicates with each other. This part will define the connection between two entities in our context diagram as a separate work area. In these work areas we specify functional requirements for each of them.

The functional requirements define what the system is supposed to do with the data that is previously defined in the data requirements section.

6.3.1 Work Area: 1. Satellite - Vultus Engine

Task 1.1 *Receive satellite data*

Purpose:	Collect data for processing
Trigger/Precondition:	Data is needed. A notification is broad casted by the satellite image holder
Frequency:	At request of user

Satellite data in the form of images is needed for analysing and building a model over the farm area. The images is used for finding anomalies in the farmland and this helps detecting plagues, pests etc.

Requirements

1.1.1	Data space needs to be available in order to download images and process them.

6.3.2 Work Area: 2. Vultus Engine - Anomaly Detection

Task 2.1 *Analyse data and train AI*

Purpose:	Find anomalies
Trigger/Precondition:	Task: 1.1, 4.1
Frequency:	Approx. every 5 days

Copernicus provides fresh satellite data every 5 days, which means new data for the Anomaly Detection System to train on. This training is to be done in conjunction with farmer data from the FMS.

Requirements

2.1.1:	The anomaly detection should start to create a model for a farmer from the moment that the farmer enters his farmland coordinates into the FMS.
2.1.2:	When training the model for the anomaly detector it should be trained from the earliest possible point where all needed data is available until the current date.

Task 2.2 *Anomaly Classification*

Purpose:	Classify anomaly severity
Trigger/Precondition:	A new satellite image is available. Task 2.1
Frequency:	Varying, dependent on the image update frequency

Each time an anomaly is detected its severity has to be classified in order to let the end user know the severity of the anomaly.

Requirements

2.2.1:	The anomaly detector should be able to find and classify datasets every time a new satellite image is available.
2.2.2:	The anomaly detector should be able to grade each dataset on different severity levels based on the anomaly.
2.2.3:	The anomaly detector should be able to classify a given farmland area.

6.3.3 Work Area: 3. Anomaly detection - User

Task 3.1 *Alert the farmer of anomaly*

Each time an anomaly is detected, this anomaly should be classified by the anomaly detection and alert the farmer to the problem and the severity of the problem detected. This so that the farmer can react quickly to the problem and prioritize if there is more than one anomaly

Purpose:	Alert the farmer of a potential problem
Trigger/Precondition:	Task 2.2
Frequency:	Varying

Requirements

3.1.1	Anomaly is notified to the FMS with the data structure described in 6.2.2.2.

Task 3.2 *Data visualization to the user*

Purpose:	Show the user the properties and location of the anomaly
Trigger/Precondition:	Anomaly classified. Task 2.2
Frequency:	Varying

When an anomaly is detected the problem should be presented to the user with sufficient and easy visualization. This so that the user can understand where the problem is and why the system have detected an anomaly. The user can also visualize the anomaly history for his farmlands.

Requirements

3.2.1:	An detected anomaly of a users farmland should be presented in the form of a heat map overlayed on top of graphical interface of the farmland.
3.2.2:	The user should be able to view the history of anomaly of his farmland.
3.2.3:	The history of the anomaly should be presented as an heat map for the user for a given point in time.
3.2.4:	The history of the anomalies should be able to be shown as an animation between 2 time intervals.

6.3.4 Work Area: 4. Vultus Engine - FMS

Task 4.1 *Deliver Farmer Data to Vultus Engine*

Purpose:	Deliver Farmer Related Data
Trigger/Precondition:	New data is available
Frequency:	Approx. every 5 days

The FMS has information about each specific farmer. This can be used to increase the AI's accuracy.

Requirements

4.1.1:	The Anomaly Detection should be able to use data from FMS to increase its test accuracy.

6.4 Use Cases

6.4.1 Farmer Checks Anomaly History

Primary Actors:

- Farmer
- FMS
- Vultus Engine

Preconditions:

- Task 1.1
- Task 2.2
- Task 3.2

Event Flow:

1. The farmer navigates to the Anomaly History section of the FMS
2. The farmer requests anomaly history for a specific time frame
   (a) Vultus Engine sends the data for the given time frame
   (b) No data for the given time frame is available

6.5 Quality Requirements

The following section will define quality requirements for the system.

6.5.1 Notifying the farmer of an anomaly

Primary Actors:

- Farmer
- FMS
- Vultus Engine

Preconditions:

- Task 1.1
- Task 2.2
- Task 4.1

Event Flow:

1. The FMS receives an anomaly alert from Vultus Engine
2. The farmer receives an anomaly notification

   (a) The severity of the anomaly is low, in which case the notification will only be available when logging into the FMS
   (b) The severity of the anomaly is moderate, in which case the farmer will be notified by email and in the FMS
   (c) The severity of the anomaly is high, in which case the farmer will be notified by SMS, email and in the FMS

6.6 Quality Requirements

The following section will define quality requirements for the system. Quality Requirements are requirements that specify how well the system must perform its functions. Quality requirements can be divided into different types of requirements, these types are: Capacity, accuracy, Performance, Usability, Security and maintainability. Quality requirements are made out from a couple of factors where different projects value the factors differently. The different factors are specified in a "Quality Grid". The quality grid is a table where you can see the relationship between the values of different factors.

Capacity Requirements
Quality requirements related to the volume of computer resources used by the product. For example memory space and computing power.

Accuracy Requirements
Quality requirements related to the accuracy of the data the product uses. Accuracy refers to the range and the precision of the data.

Performance Requirements
Quality requirements that specify how fast the product shall be. This can for instance be the response time for the functions of the product.

Usability Requirements
Quality requirements related to the usage of the program. For example how easy it is to use.

Security Requirements
Quality requirements related to the security of the system. For example, many software products should often have high security since they might contain sensitive information about users.

Maintainability Requirements
Quality requirements related to the maintainace of the product. For example repair and extending the product.

Quality factors Vultus	Critical	Important	As Usual	Unimportant	Ignore
Integrity/Security			X		
Correctness	1				
Reliability/Availability			X		
Usability	2				
Efficiency				X	
Maintainability			X		
Testability		3			
Flexability				4	
Portability					X
Interoperability		5		6	
Reusability					X
Installability					X

Table 2: Quality Grid.
1. False positives are okay, but false negatives are never okay.
2. Our users are not the most technical people.
3. Testability are important to be able to support really good correctness.
4. Vultus engine already supports flexible modules.
5. Communication with Vultus engine are important.
6. External communication is unimportant.

References

[1] Vultus `http://www.vultus.se` Fetched: 2017-11-07.

[2] Copernicus, Europe's eyes on Earth. `http://www.copernicus.eu` Fetched: 2017-11-07.

[3] Computers and Electronics in Agriculture. *Farm management systems and the Future Internet era* 2012 Elsevier B.V.

[4] European Commission. *IN-DEPTH REPORT Nitrogen Pollution and the European Environment Implications for Air Quality Policy* `http://ec.europa.eu/environment/integration/research/newsalert/pdf/IR6_en.pdf` Fetched: 2017-11-18

[5] Internet Engineering Task Force. *The GeoJSON Format* `https://tools.ietf.org/html/rfc7946` Fetched: 2017-11-18