University of Groningen

Software Architecture

Group 3

Smart Flood Monitoring (SFM)

Authors:

Eedema, Gerrit
Putra, Guntur
Fakambi, Aurélie
Schaefers, Joris
Brandsma, Jeroen
Menninga, Wouter
(Klinkenberg, Guus)

Version: 0.7
18 Oct, 2015

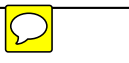
Authors

Name	E-Mail
Eedema, Gerrit	G.Eedema@student.rug.nl
Putra, Guntur	G.D.Putra@student.rug.nl
Fakambi, Aurélie	A.Fakambi@student.rug.nl
Schaefers, Joris	J.Schaefers@student.rug.nl
Brandsma, Jeroen	J.T.Brandsma@student.rug.nl
Menninga, Wouter	W.G.Menninga@student.rug.nl
(Klinkenberg, Guus	G.Klinkenberg@student.rug.nl)

Revision History

Version	Author	Date	Description
0.1	all	06-09-15	Setting up the working environment and the initial document structure. Draft versions of the first four chapters created, which are (respectively) Context, Architectural business information, Requirements and Analysis.
0.2	Menninga, ...	10-09-15	First version of functional requirements
	Schaefers & Eedema	13-09-15	Changed the context and enhanced the business information
	Brandsma	13-09-15	Updated the requirements part
	Putra	13-09-15	Updated the stakeholder of requirements part
0.3	Menninga	17-09-15	Updated the functional requirements
	Brandsma	17-09-15	Updated architectural vision and use-cases
	Klinkenberg & Eedema	17-09-15	Updated technical requirements, some formatting, started work on the first two decisions
	Menninga	19-09-15	Added and updated functional requirements
	Putra	19-09-15	Processed feedback, revised the stakeholder, and added decision on database selection.
	Eedema	19-09-15	Processed feedback
		19-09-15	Improved business chapter
	Menninga	20-09-15	Updated assumptions and improvements to structure of analysis chapter. Added decision about type of water level sensor.
	Brandsma	20-09-15	Updated requirements chapter. Added decision about cloud computing.
	Schaefers	20-09-15	Improved business chapter and processed feedback. Checked the use cases and put them in tables.
0.4	Menninga	23-09-15	Improved functional requirements after feedback
	Brandsma	24-09-15	Revised architectural vision, extended and improved use-cases
	Menninga	24-09-15	Changed layout of the risk assessment section
		24-09-15	Changed layout of the risk assessment section
	Fakambi	24-09-15	Improvements to the non-functional requirements
	Eedema	24-09-15	Improvements to the business part
	Putra	24-09-15	Improved stakeholder with new quality attributes. Added key-drivers after stakeholder.
	Menninga	26-09-15	Complemented and improved non-functional requirement section Removed some obsolete functional requirements Improved risk assessment section
	Putra	26-09-15	Business rationale.
		27-09-15	Draft chapter 5,6, and 7.

	Brandsma	27-09-15	Improved some figures. Improved chapter 4. Reviewing.
0.5	Putra	30-09-15	Hardware architecture draft.
0.5	Menninga	30-09-15	Changed Use Case diagram. Improved and added use cases Added High Level Requirements Updated stake holders (added weights) and changed key drivers accordingly Small improvements to the non-functional requirements Changed the evolution requirements Added a risk about 3rd parties not developing apps
	Eedema	01-10-15	Worked on chapter 5
	Brandsma	01-10-15	Worked on chapter 5
0.5	Menninga	03-10-15	Added decision about dike sensors (and connectivity)
0.5	Menninga	04-10-15	Small updates to chapter 3, added glossary
	Putra	04-10-15	Added decision on main analytics servers, database servers, and switches Added hardware description of main analytics servers and database servers.
	Menninga	04-10-15	Changed hardware overview diagram
0.6	Menninga	08-10-15	Updated hardware costs ch. 2 + improvements ch. 3 Added activity diagram for flood monitoring / warning
	Eedema	08-10-15	Updated chapter 5 + logical view
	Brandsma	08-10-15	Updated chapter 5 + logical view
	Putra	08-10-15	Updated figures of cluster on chapter 6. Added deployment view of software architecture.
	Menninga	11-10-15	Updated use case about warning citizens Added sequence diagrams for sending data and warnings Updated dikes sensors information
	Putra	11-10-15	Updated figures of cluster on chapter 6.
		11-10-15	Updated figure of hardware overview on chapter 6.
		11-10-15	Updated figure of deployment view diagram on chapter 7.
	Eedema	11-10-15	Setup chapter 8.
	Schaefers	11-10-15	Enhanced logical view
		11-10-15	Added decision table for Arduino.
	Menninga	11-10-15	Added Push/Pull decision.
	Fakambi	11-10-15	Hardware decision and description for UAV.
	Brandsma	11-10-15	Updated chapter 5
0.7	Putra	15-10-15	Updated deployment diagram. Added artifacts in deployment diagram.
	Menninga	15-10-15	Added sensor communication decision
	Brandsma	15-10-15	Improved chapter 6
	Menninga	18-10-15	Adding ATAM tables
	Putra	18-10-15	Added Technical non-functional requirements verification (performance, interoperability, security, scalability).
	Schaefers	18-10-15	Updated component diagram, enhanced logical view
	Eedema	18-10-15	Creating and enhancing implementation



Contents

Revisions . . . i
Table of Contents . . . ii
List of Figures . . . vi
List of Tables . . . vii
Glossary . . . vii

1 System Context **1**

2 Architectural business information **2**
2.1 Business opportunity . . . 2
2.2 Business rationale . . . 2
2.3 Product and service description . . . 3
2.4 System in the market . . . 4
2.5 Financial model . . . 5
2.5.1 Software Architecture costs . . . 5
2.5.2 Development costs . . . 5
2.5.3 Hardware costs . . . 6
2.5.4 Total costs . . . 7
2.6 Competitors . . . 7

3 Requirements **8**
3.1 Architectural vision . . . 8
3.2 Stakeholders and their concerns . . . 9
3.3 Key-drivers . . . 10
3.4 High-level requirements . . . 11
3.5 Stories and use-cases . . . 12
3.5.1 Receive sensor data . . . 12
3.5.2 Receive weather/GEO data . . . 14
3.5.3 Subscribing to SMS Service . . . 15
3.5.4 Determining flood probability . . . 16
3.5.5 Warn citizens in case of an imminent flood . . . 17
3.5.6 Warn safety region in case of an imminent flood . . . 18
3.5.7 Third party accessing data through the systems API . . . 19
3.5.8 Detecting a faulty sensor . . . 20
3.5.9 Maintenance employee checks system state . . . 21
3.6 Functional requirements . . . 22
3.7 Commercial non functional requirements . . . 23
3.8 Technical non-functional requirements . . . 23
3.8.1 Reliability . . . 23
3.8.2 Availability . . . 23
3.8.3 Resilience . . . 23
3.8.4 Performance . . . 24
3.8.5 Interoperability . . . 24
3.8.6 Security . . . 24
3.8.7 Scalability . . . 25
3.9 Evolution requirements . . . 25
3.10 Risk assessment . . . 26
3.10.1 Technical . . . 26
3.10.2 Business . . . 27
3.10.3 Schedule . . . 28

4 Analysis **29**
4.1 Assumptions . . . 29
4.2 High-level Design Decisions . . . 30

5 System architecture **34**
5.1 System context . . . 34
5.1.1 Diagram . . . 34
5.1.2 Users and Roles . . . 34
5.1.3 External Systems . . . 35
5.1.4 Channels and Information Flows . . . 35
5.1.5 Alternatives . . . 36
5.2 Verification . . . 37
5.2.1 Availability . . . 37
5.2.2 Cost . . . 37
5.3 Elaborated Model . . . 37

6 Hardware Architecture **39**
6.1 Hardware Overview . . . 39
6.2 Hardware Design Decisions . . . 40
6.3 Hardware Description . . . 47
6.3.1 Sensor Components . . . 47
6.3.2 UAV's . . . 47
6.3.3 Database Cluster and Data Collection . . . 48
6.3.4 Analytics Components . . . 48

7 Software Architecture **50**
7.1 Software architecture design . . . 50
7.2 Architectural view . . . 50
7.2.1 Scenarios . . . 50
7.2.2 Logical View . . . 51
7.2.3 Implementation view . . . 60
7.2.3.1 Components . . . 60
7.2.3.2 Packages . . . 62
7.2.4 Process View . . . 64
7.2.5 Deployment View . . . 65
7.2.5.1 Deployment Diagram . . . 65
7.2.5.2 Artifacts . . . 68
7.2.6 Database . . . 69
7.3 Software design decisions . . . 71

8 Architecture evaluation **75**
8.1 Requirements validation . . . 75
8.1.1 Functional requirements . . . 76
8.1.2 Commercial non-functional requirements . . . 76
8.1.3 Technical non-functional requirements . . . 77
8.1.3.1 Reliability . . . 77
8.1.3.2 Availability . . . 77
8.1.3.3 Resilience . . . 77
8.1.3.4 Performance . . . 77
8.1.3.5 Interoperability . . . 77
8.1.3.6 Security . . . 77
8.1.3.7 Scalability . . . 78
8.2 Architecture evaluation . . . 79
8.2.1 Architectural approaches . . . 79
8.2.2 Quality attribute utility tree approaches . . . 79
8.2.3 Scenarios . . . 80

9 System evolution **83**

Appendices **84**

A Time Tracking **84**
A.1 Week 1 . . . 84
A.2 Week 2 . . . 84
A.3 Week 3 . . . 85
A.4 Week 4 . . . 85

A.5 Week 5 85
A.6 Week 6 86
A.7 Week 7 86
A.8 Todo 86

List of Figures

3.1 Schematic overview of the flood monitoring system 8
3.2 Use-case diagram 12

5.1 System context diagram 34
5.2 Elaborated system context diagram 38

6.1 Schematic overview of the hardware architecture of Smart Monitoring 39
6.2 Logical schematic of database cluster of SFM 48
6.3 Logical schematic of analytic cluster of SFM 49

7.1 Scenario for a flood 51
7.2 Scenario for guidance 51
7.3 Layers of the software 53
7.4 Class diagram of service layer 54
7.5 A sequence diagram of sending the sensor data 54
7.6 Sequence diagram of the client pushing the sensor data 55
7.7 A sequence diagram of sending warnings to citizens and emergency room 56
7.8 Class diagram of the domain layer concerning REST calls 57
7.9 Class diagram of flood prediction in the domain layer 58
7.10 Component diagram 61
7.11 Package diagram 62
7.12 An activity diagram of the flood monitoring process 64
7.13 Deployment diagram 67
7.14 Database diagram 70

List of Tables

2.1 SWOT analysis of SFM . . . 3
2.3 Approximation of development costs . . . 5
2.5 Total costs of development . . . 6
2.7 Total costs of development. UoM=Unit of measurement . . . 6
2.9 Total hardware cost . . . 6
2.11 Total cost of the system . . . 7

3.1 Quality attributes of Software Architecture from "Software Requirements" Book [25]. . . . 9
3.2 Matrix of stakeholders concern. . . . 10

4.1 Decision – Operating system . . . 30
4.2 Decision – Connectivity of the sensors . . . 31
4.3 Decision – Type of water level sensors . . . 32
4.4 Decision – Server park . . . 33

6.1 Decision – Choice of Sensors . . . 40
6.2 Decision – Connectivity of the dike sensor . . . 41
6.3 Decision – Arduino selection . . . 42
6.4 Decision – UAVs . . . 43
6.5 Decision – Analytic cluster selection . . . 44
6.6 Decision – Choice of storage machine. . . . 45
6.7 Decision – Choice of high performance switch . . . 46
6.8 Hardware specification of Dell PowerEdge R530 . . . 49

7.1 Artifact for sensing component . . . 68
7.2 Artifact for main analytics cluster . . . 69
7.4 Decision – Data source architectural pattern . . . 71
7.6 Decision – Database . . . 72
7.7 Decision – Linux Distribution . . . 73
7.8 Decision – Push / Pull for sensor data . . . 73
7.9 Decision – Securing sensor data communication . . . 74

8.1 ATAM – Handling faulty sensors . . . 80
8.2 ATAM – Handling hardware failures . . . 81
8.3 ATAM – Ensuring availability of third party data / services . . . 82

Glossary

API Application Programming Interface.

SFM Smart Flood Monitoring.

UAV Unmanned Aerial Vehicle.

1 System Context

We are the architecture team of the company RugSAG3. Our team is specialized in making smart systems for environmental monitoring.

All around the world, natural disasters cause a lot of trouble. These disasters can result in catastrophic events that cause deaths and require a huge amount of money to repair the damages caused. People lack the knowledge on when the disaster is about to happen, how to properly prepare and how to properly act during such a disaster.

Climate change and extreme weather phenomena cause these disasters to get worse over time. Every year the amount of these disasters increases and they become increasingly more severe. This causes the damage of the natural disasters to increase, which means that there is a great need for a system that can reduce this damage and helps the people during natural disasters like these. The system that we develop aims to do this. The goal of our first product is to:

1. save lives,
2. reduce damage costs,
3. reduce the social consequences.

To achieve the aforementioned goals, the system will provide predictions with regard to upcoming floods.

The first release of this system that we, as the architecting team of RugSAG3 design, will only support floods as a natural disaster. The system will provide warnings and guidance to the necessary people before and during a flood. By using various kinds of sensors, like vehicles and control units, this system monitors certain areas. If something suspicious happens, the system will check and verify the information in order to not give false flood warnings. When a flood indeed occurs (or will occur), the system provides warnings and guidance to the necessary people during and, if possible, before a flood.

When this flood monitoring system works as planned and shows that it can indeed reduce the damage of floods, RugSAG3 will extend the systems functionality by adding support for the monitoring of other kinds of natural disasters and situations that desire a monitoring system.

Over time, the system will reduce social, financial and human losses of natural disasters, starting with floods. This will revolutionize the way we think about natural disasters. Though the market around environmental monitoring is very competitive, this system aims at being a core part of future monitoring systems by being as dynamic and flexible as possible. Thereby allowing new features to be added easily, allowing the system to grow over time, including new upcoming user needs.

2 Architectural business information

The following section describes the different aspects of the business environment of the Smart Flood Monitor. First we will explain our vision and why there is place for us at the market.

After this the product and its stakeholders will be explained. This chapter is completed with a more detailed look at the business model and some models about the market and the financial prospect.

2.1 Business opportunity

There are many natural disasters happening each year all over the world. Each year these disasters take lives, destroys a lot of properties, and cause social disturbance. Looking for example at the Indian ocean's tsunami in 2004, it looks that the damage could have been significantly reduced if the necessary people were warned[17, 16].

In the future more floods are expected because of global warming[19]. The rise of the sea level is a consequence of global warming. Another consequence is the increase of extreme weather events, for instance heavy rainfall. It is expected that natural disasters will cause 300 billion in losses annually in the upcoming decade[18]. This justifies to invest a high amount of money to minor the losses of a flood. Not only the losses in terms of money, but more importantly lives, and also social impact.

The increasing thread of the floods is the basis for our vision. RugSAG3 has the following vision: No people in the world will be harmed by floods. We want to find solutions so floods does not take lives, cause social consequences, and damage properties.

The mission of RugSAG3 is to design a flood warning system in order to save lives and reduce costs. A next generation reliable flood monitoring and warning system will help lower the catastrophic impact of floods. The system will predict imminent floods and send out people in order to reduce the impact of a flood. In the future the system should be able to monitor other kinds of disasters and send out warnings.

The Netherlands is a country that is situated for large parts(26%) under the sea level[2]. Using dikes and other solutions they protect their country against the water. The increasing sea level causes extreme danger in the Netherlands. It is important that they can monitor how dangerous imminent floods are and take action if things are looking to wrong. These properties of the Netherlands make it an ideal market to start deploying our system.

2.2 Business rationale

RugSAG3 will develop a new smart flood warning system to minimize the damage caused by floods. As previously mentioned, in the Netherlands the protection against floods is an important issue, because major part of the country itself is actually below sea level. Global warming will increase the urgency of this issue. Thus, the people within the Netherlands must be seriously protected against floods. The flood warning system will detect floods, warn people and governmental institutions located in the disaster area and provide guidance to the safety region.

RugSAG3 is a new player within the flood warning system domain. However, to gain significant market share and to compete against the existing player in the market, RugSAG3 will launch a reliable and adequate product that uses the newest technologies by using sensors and automated systems. RugSAG3 will also use hardware which is already on the market and is already tested as well. Buying third party hardware will also speedup the development of the system and lower the development costs.

It is important to evaluate the strengths and weaknesses of RugSAG3. Using this evaluation, we can reflect ourselves with respect to the competitors, which will lead to a better understanding of the opportunities and threats RugSAG3 owns in the market. We map our position on the market by using SWOT-analysis. This analysis maps our strengths, weaknesses, opportunities, and threats. The results of our analysis is shown in Table 2.1.

Strengths	Weaknesses
• Adjustable system that is future proof • Having a low selling price. Lower then the competitors • Diverse team with several skills in IT and management • Having a good management team • Frequent discussion with technical and business experts in the field. • Experience with working with sensors • Diverseness of cultures in the development team	• Complexity of decision making • New project team • No experience with creating environmental monitoring systems • No domain knowledge of floods
Opportunities	**Threats**
• Due to climate change, the market will grow and such a system becomes more urgent • Other kinds of natural disasters are threatening, which could be monitored • Smart sensors are a hot topic, new sensors will be developed. Making the system support and use the newest sensors allows it to obtain more, and a wider variety of valuable information	• High hardware costs • New competitors will enter the market because of a growing market • Climate change will force the system to be improved over time • External parties that affect our system could change or stop

Table 2.1: SWOT analysis of SFM

The initial product price will be low in order to get a market share and prove the product in a real-time environment. By providing maintenance and updates in the future RugSAG3 will earn money to improve the product further and sell it to other potential customers. This in combination with the increasing need of a reliable warning system for imminent floods will result in a viable business. To extend the capability and the reliability, the system is also extendable with numerous additional and optional features.

To sum up, the unique selling points of our system are:

- Low initial product cost. This means customer can afford this system in a low price with basic functionality.
- High system extendability. Although the initial offer to the customer is a basic product, the system is actually highly extendable with additional support and feature

Citizens will get a notification in case of imminent flood — they will either receive notifications from the government or from the system directly through subscription. In this way, people will know when a certain area will be flooded, or when the flood is about to happen. These are some of unique features of our system, which will also drive the system to be successful. The main goal of this project, however, is to safe lives, reduce costs and reduce social consequences. These main goals will be met when:

1. 99% of the people in a dangerous area regarding a flood, receive a warning message. This message must contain enough information for receivers to know whether they are save or not and if not, how they can get to a save location.
2. 99% of the people who receive a warning successfully get to a safe environment in time.
3. 99% of the people receiving information before or during a flood, find these messages helpful and reported that it guided them successfully in order to save extra lives and/or goods.

When the first version of Smart Monitoring is released and is used to start monitoring actual floods, its success will be measured according these statistics. Getting a warning message to the people who requested to be warned is the most important thing to do. Using this warning message, people can move to a saver location.

2.3 Product and service description

RugSAG3 offers a flood warning system. When a imminent flood is monitored by the sensors of the system a warning should be sent to governmental organizations and people within the danger area. Also a possibility of

guidance should be provided when a flood is happening to assist rescuers and guide inhabitants to safe areas. The guidance part will be done by third party application developers. The system will publish the data that these developers can use to build their service.

Basically the system will consist of four subsystems: monitor the state of dykes and water levels, analyze the data from the monitoring part to detect imminent floods, warn governmental organizations and inhabitants in the danger area, and provide data which third party developers can use to build applications which help search and rescue organizations and inhabitants for guidance.

The first subsystem (monitor the state of dykes and water levels) will consist of various sensors that are placed near and in dykes and water ways. The state of the dikes must be monitored continuously, i.e. pressure of the dyke. Also, sensors must be installed to monitored continuously the water level. The data of all the sensors will be sent to a server, in a safe location, to store all the data.

The second subsystem (analyze the data from the monitoring part to detect imminent floods) will analyze all the data from the sensors and data from weather forecasting service. Based on this data an algorithm will monitor continuously if there are dangerous situations.

The third subsystem (warn governmental organizations and inhabitants in the danger area) will send warning messages when the algorithm identified a dangerous situations. The safety region will receive a warning message that an area is in danger. Information like position, area, sort of danger and amount of danger will be send. The safety region will be responsible to take action based on this information. Inhabitants can receive warnings via sirens, mobile phone, radio, television.

The last subsystem (provides data to third party developers) will publish data that other companies can use to build applications. The data that will be published is mainly for provide guidance for emergency services and people which are in the dangerous area.

The service will consist of maintenance for the product and upgrades.

As described before, floods will be the first natural disaster the initial system supports. The system will be based on an architecture that other kind of disasters that can be monitored by sensors can be easily implemented. Natural disasters that can be monitored in the future by the system are: Volcano eruption, earthquakes, and slides. By implementing sensors and algorithm's for these natural disasters our system will become more viable for in the future. In figure there is an overview of the implementation of natural disasters over time.

Business Plan , Roadmap and expansion of RUGSA3

First release 2016 NETHERLANDS
2016-2018:
Still focusing on flood
New features : Building our own application
Adding external input
Maintenance
Statistics about reliability
New sensors, New UAV's => more accuracy and follow the new techonologies
2018-2025 : Expansion to Europe starting by countries which have high risk and history of floods (documentation , studies in order to choose countries which really need our product , countries where this market doesn't exist yet or at least isn't in expansion => good business strategy) => more production
Profit
Big data , new algorithm
2025-2030 Expansion to the world starting by countries which have high risk and history of floods (documentation, for instance Asia)
Work on others natural disasters like earthquakes , volcanoes.

2.4 System in the market

All over the world floods are causing tremendous trouble to people. This makes that our system can be sold all over the world. First the system focuses on the Netherlands. When the system is working correctly other governments in the world must be informed about the system. First focusing on Europe and than other continents.

Third party application developers have the opportunity to build application that enables guidance for emergency services and people. First the focus is on finding one company that has experience with building such products. Then other software companies should get noticed about the system. The more third party applications are build, the better system will reach it's mission.

Safety regions are responsible when disasters are happening in the region. They consist of local governments, emergency services, the army, and water boards. Safety regions and the organizations that are part of the safety regions should be informed about the system. When they know about the system they can inform the government and put pressure to buy this system.

When a flood is happening and things go wrong inhabitants will ask governments what went wrong and state that they want to be better protected. They will pressure governments that they have the best flood warning system.

2.5 Financial model

The financial model will be a low product price. This in order to price the product low in the market. A service description for maintenance will be offered. Also updates will be sold to the customer

2.5.1 Software Architecture costs

The software architecture team of RugSAG3 consists of six members. Creating the architecture of the project is estimated to take ten weeks. All team members will spend 15 hours a week on the project. This totals $6 * 10 * 15 = 1050$ working hours. Each working hour costs €150,-. Total spend on the software architecture is €157.500

2.5.2 Development costs

The costs of developing the system is approximated in table 2.3, shown below.

Description	**Man hours**
Get values from various sensors	160
Get weather forecasts	160
Flood prediction algorithm	3100
Warning messaging system	2000
Guidance information system	1500
Redundancy and fail over systems	1000
Testing & debugging	600
Release build	250
Overhead	1000
Total hours	9770

Table 2.3: Approximation of development costs

Development of the system comprises the development of several different aspects. For each of these aspects, table 2.3 shows an approximation of the hours needed for the development. Resulting in a total amount of 9770 hours needed to create the system.
The development is done by RugSAG3 self. Each member of the project team assigned to develop this system is paid €50 an hour. This results in a development cost of $50 * 9770 = 488.550$. As shown in the table below.

Description	Cost
Hours	9770
Cost per hour	€50
Total cost	€488.500

Table 2.5: Total costs of development

2.5.3 Hardware costs

Around 17.000 kilometers of dikes protect the Netherlands against flooding [3]. The monitoring system consists of GeoBeads sensors with a claimed life time of 10 years, which are installed with conventional push-in techniques.

The GeoBeads sensor costs about 350€ per unit. For every 3 meter of dike, one such sensor is needed. In the Netherlands, there are about 17.000 kilometers of dike. However, it is estimated that only about 2500km of the dikes have to be monitored by dike sensors (because the water board considers these dikes potentially unstable).

Each push-in costs about €200 and three sensors can be installed per push-in[23].

The following table gives an overview of the costs related to the dike sensors:

Description	Value	UoM
Total length of all dikes to be monitored	2.500	km
Number of GeoBeads per kilometer	330	
Sensor cost	350	€
Installment cost per cross-section	200	€
Sensors per cross-section	3	
Base installation cost factor like labor and wiring	0.2	
Life expectancy of GeoBeads	10	years

Table 2.7: Total costs of development. UoM=Unit of measurement

This makes the costs for the dike monitoring as follows:
Initial cost per km: $330 * 350 + (330/3) * 200 = 137.500€ * 1.2 = 165.000€$
Total initial cost: $137.500 * 2.500 = 412.500.000€$

The yearly maintenance costs for the dike sensors is estimated to be around 4.500.000€ per year. If the depreciation of the GeoBead sensor is added, this becomes $4.500.000 + 33 * 825.000 = 31.725.000€$.

The system then needs several servers:

- Sensors
- State monitor server
- Danger check server
- Warning system server
- Guidance server

There are four servers and each server costs €1.500. To enhance the availability, the servers will be redundantly set up four times. Resulting in a hardware cost for the servers of
Server cost= $4 * 1500 * 4 = 24.000$. The total hardware costs is shown in the table below.

Description	Cost
Installation costs	€1.683.000
Maintenance costs	€336.600
Server costs	€24.000
Total cost	€2.443.600

Table 2.9: Total hardware cost

2.5.4 Total costs

The total costs for the system is calculated in the table below.

Description	Cost
Software Architecture costs	€157.500
Development costs	€488.500
Hardware costs	€2.443.600
Total cost	€3.089.600

Table 2.11: Total cost of the system

So the total costs of the system is €3.089.600

2.6 Competitors

Siemens is a competitor that already has developed a flood-warning system in Belgium in 2006. When the system detects an imminent flood it sends a SMS to people that live near the rivers in order to warn them. The system is implemented in a small region, which consists of only 3 rivers. The total costs for this system were €230.000. Later on they continued engineering the system. The system uses sensor which measures: temperature, water pressure, and shifting. They participated with this system in the Urban Flood project from the EU. The strong point of this competitor is that they already have funding and also experience with building such a system. Siemens is the main competitor for RugSAG3. Their strength is that they already have a proven system in Belgium and have a lot of experience through participating in flood monitoring research projects. The opportunity for RugSAG3 is to engineer a flexible system that can be used in the future for other kind of disasters. Further on by selling the product at a low selling price and profit on the maintenance work and upgrades the Netherlands will be interesting in using the product.

Further there are Universities that do research on flood warning systems. At the Malaysian Institute Information Technology they developed a product prototype of a system called Intelligent Flood Information System via SMS. Water level sensors are the only one they used. When there is a flood they send a warning SMS to people that are within the area. In the future this could become a competitor if they create a start-up or sell the idea to a company.

3 Requirements

This chapter describes our vision and uses it to derive stakeholders. This information helps us to be able to properly write use cases and stories. These will be used to extract functional, commercial, technical and evolution requirements. Afterwards, a risk assessment will take place, to ensure that the project is not at great risk.

3.1 Architectural vision

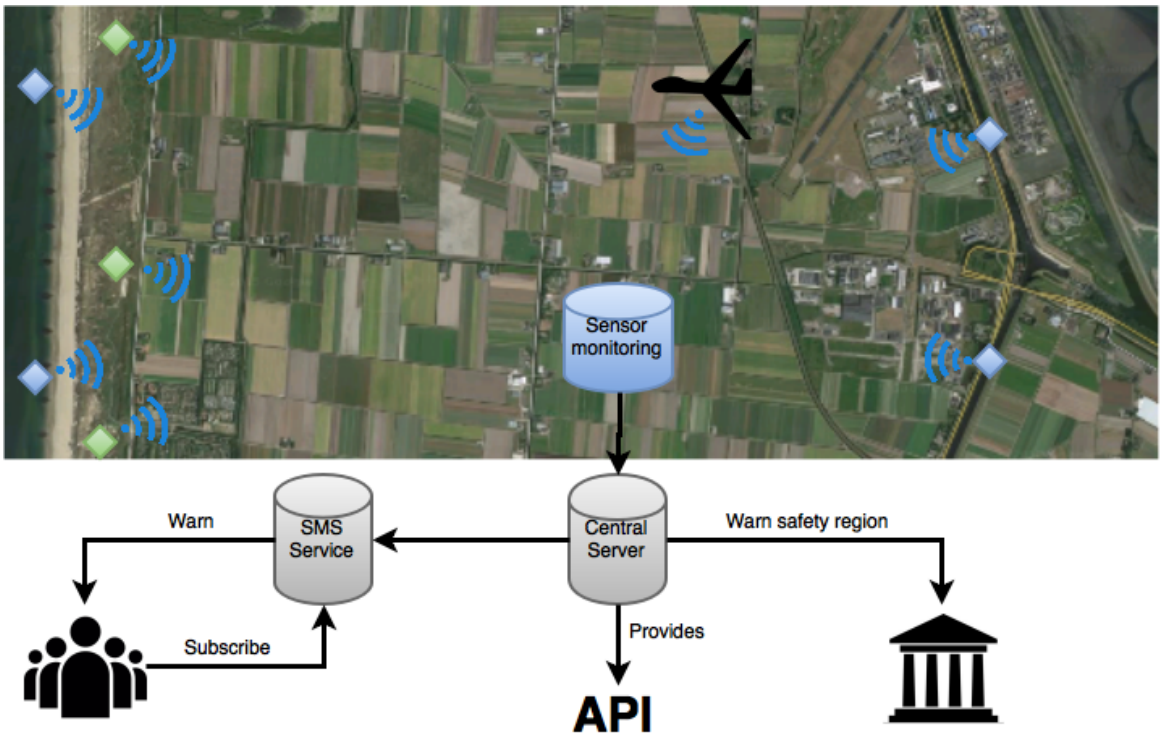


Figure 3.1: Schematic overview of the flood monitoring system

The SFM system consists of multiple parts. These are represented in figure 3.1. First of all there is the monitoring part. This part monitors the current state of the environment. To achieve this, we need a lot of data, which we obtain using sensors, weather APIs and UAVs. We use sensors to get the current water level of waterways, these water sensors are shown in the figure as blue squares. We also measure the density and structure of dikes, this is done by pressure meters, temperature meters and tilt meters. These are represented as green squares. To see how far a flood has spread we use UAVs. The data that is obtained through all sensors will be send to the central server. Here the information will be processed. The system then determines if there is an imminent flood.

In case of an imminent flood a warning will be issued to the government and the citizens who live in the threatened area. We do this by issuing a warning to the government. In their turn the government uses their infrastructure to warn the citizens. In the Netherlands this infrastructure consists of a siren system and an SMS-system. Besides this people can also apply for our SMS service. People who are subscribed to this service will receive a text message with more information about the imminent flood.

If people want to have more information on an imminent flood, they can always access the API. This API gives relevant raw data about imminent floods. To make a usable interface for the citizens, we want to cooperate with third party developers. These developers could create an application that can guide citizens to a save area.

3.2 Stakeholders and their concerns

This section defines all stakeholders of our system and describe the concerns of the stakeholders. A stakeholder might be a person, group of persons, or organization that are involved in our system. There are eight stakeholders, ranged from first parties to third parties stakeholders. We use several quality standards from "Software Requirements" book by Microsoft [25]. Those quality standards are described in Table 3.1.

Quality Attributes	**Brief description**
Availability	The extent to which the system's services are available when and where they are needed
Interoperability	How easily the system can interconnect and exchange data with other systems or components
Performance	How quickly and predictable the system responds to user inputs or other events
Reliability	How long the system runs before experiencing a failure and how reliable the results of the system are (accuracy).
Security	How well the system protects against unauthorized access to the application and its data
Usability	How easy it is for people to learn, remember, and use the system

Table 3.1: Quality attributes of Software Architecture from "Software Requirements" Book [25].

There are six quality attributes, as can be seen in Table 3.1, for measuring stakeholders' concern regarding our system. Furthermore, we also add profitability as another quality standard to improve measuring stakeholders' concern. Detailed description of stakeholders and their concerns are explained below.

Product owner is concerned about the reliability and profitability of the system. The product owner funds the whole project and is highly concerned about the profitability. Thus, to gain big market share and extract large profit from this product, the product owner has to make this product reliable.

Developers are concerned about interoperability, performance and security. We, the architect team of RugSAG3 company, are also part of this. This stakeholder is responsible for the development of the systems until it is ready for production. Including architecting, designing, analyzing, testing and implementing this SFM System.

Third party developers are concerned about interoperability, availability, usability and reliability. Third party developers are important for our system since they need to provide an application that will give the users guidance in case of a flood.

Competitors are concerned about reliability and profitability. Competitors give negative effect on the system because competitors will be aiming on the same customer target. On the other hand, competitors are also triggering us to make a really good system in order to be able to compete with them and to save more lives. Thus, competitors must also be kept in consideration.

Government is concerned about availability and reliability. The government will be the main customer of this product, specifically, The Dutch Ministry of Infrastructure and the Environment. The government will be part of mitigation when the flood is imminent. This system will help the government by notifying them when it detects a flood and supplying them with relevant information about the flood.

Citizens are also concerned about availability and reliability, but also usability, since they can subscribe to warning by SMS. The Dutch residents are indirect user of this systems. Furthermore, they want this system to always be available and run correctly and notify them with reliable information.

Insurance companies are concerned mostly about performance, reliability and availability. The damages caused by flood sometimes are also covered by the insurance companies. Thus, the insurance companies will also be part of the stakeholders and they will make sure that their business is running well.

Local companies are concerned about availability and reliability. Local companies will also be affected by the flood, since they have a lot of resources which are in danger. Local companies want to know whether

or not this system is reliable so that they can arrange a proper action set when the flood comes to save their assets.

Safety region is responsible for the emergency services and is concerned about interoperability, performance, availability, reliability and usability. Emergency services are important when any accident happens, including flood. They will be really concerned about the thing that makes this system reliable, and inter-operable to their current system.

Table 3.2 illustrates the stakeholder concern matrix. In our approach every stakeholders are equally the same. Thus, each stakeholder receives 100 points in total that has to be distributed among all the concerns.

		Concerns						
Stakeholder	Weight	Availability	Interoperability	Performance	Reliability	Security	Usability	Profitability
Product owner	1				60			40
Developers	1		40	30		30		
Competitors	1				40			60
Government	2	60			40			
Citizens	2	40			40		20	
Insurance companies	1	35		15	50			
Local companies	1	60			40			
Safety region	3	20	15	20	30		15	
Total		355	85	105	440	30	85	100

Table 3.2: Matrix of stakeholders concern.

As can be seen from Table 3.2, the most important concern of our system is the reliability, following availability as the second most important concern. This is also identical with our significant key driver.

3.3 Key-drivers

Table 3.2 has shown the concerns of the stakeholders of our system. Not every stakeholder is equally the same in our system. Each stakeholder receives 100 points to be distributed to all quality attributes, but is also assigned a weight. By doing this, important quality attributes can be determined and those will be the key-drivers of our system.

It is clear that **reliability** is by far the most important quality attribute of our system, followed by **availability** as the second most important, and **performance** as the third most important quality attribute. Thus, those three quality attributes will be the *key-drivers* of our system. We decided to choose only three quality attributes as the key drivers for our system because choosing more than three may possibly drive our system architecture to be more complex and consume more time to be developed.

3.4 High-level requirements

The high-level requirements describe the high-level functionality of the system. The high-level requirements are used to derive the functional requirements in section 3.6. The high-level requirements are also used to classify the severity of the risks in section 3.10.

Nr.	Prio	Description
HL-1	**Must**	The system monitors the environment to determine when a flood is developing. This is the most important high-level requirement and the main feature of the system.
HL-2	**Must**	The system issues a warning to the safety region and subscribed citizens when a flood is imminent. The warning to the safety region contains relevant information about the flood, which can be used by the safety region to determine if evacuation of certain areas is needed.
HL-3	**Must**	The system can collect and supply the safety region with relevant information regarding the flood. The collected information should at least contain the location and severity of the flood.
HL-4	**Must**	To allow third-party developers to develop applications to guide citizens in case of a flood, the system exposes an API which can supply these developers with data related to the flood. This API is also used by the safety board to get relevant information about the flood.
HL-5	**Must**	There is a control panel for maintenance, where warnings etc. can be viewed and the sensor data can be viewed manually. This control panel will be used by maintainers of the system, who will be responsible for repairing/replacing sensors and system faults.

3.5 Stories and use-cases

This section will give an overview of the different use-cases. Figure 3.2 displays the use-case diagram. This provides an overview of the use-cases with their actors. In the subsections below, the architectural important use-cases are explained in more detail.

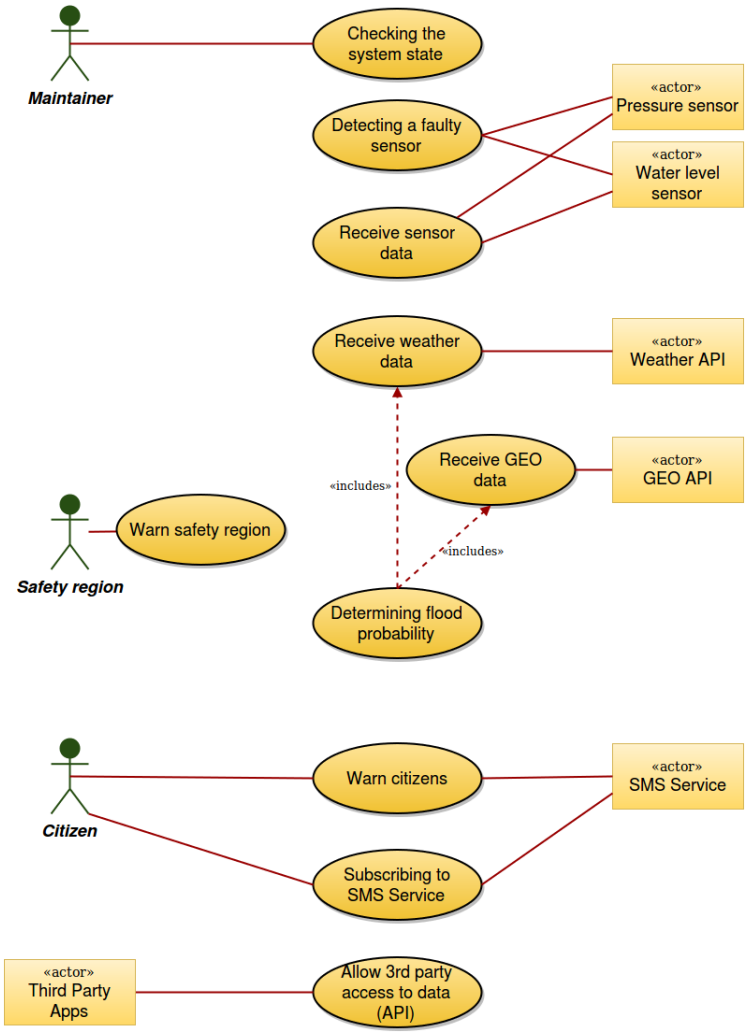


Figure 3.2: Use-case diagram

3.5.1 Receive sensor data

Number	UC-1
Description	The system receives data from the different sensors deployed
Stakeholders and interests	**Developers**: Developers need to work with the sensor data
Primary actor	System
Scope	Monitoring part of the system
Level	Sub process
Precondition	The sensor is connected to a processing unit
Main success scenario	1. The sensor does a measurement every 10 seconds 2. The sensor sends the data to the central server every minute 3. The central server normalizes the received data 4. The central server stores the normalized data to the database 5. The database stores the data

Postcondition	The database received and stored the sensor data
Alternatives	2a. 1. The data cannot be sent 2. Data will be lost 3. The use-case ends
Related requirements	FR-1, FR-3, FR-5, FR-6

3.5.2 Receive weather/GEO data

Number	UC-2
Description	The system receives data from the weather forecast service
Stakeholders and interests	**Developers**: Developers would like to have a simple to use API
Primary actor	System
Scope	Monitoring part of the system
Level	Sub process
Precondition	The system needs external weather data to predict floods
Main success scenario	1. The processing unit determines it needs forecast weather data 2. A call is made to the weather forecast service 3. The weather forecast service returns the requested data
Postcondition	The system received the forecast data
Alternatives	3a. 1. The data cannot be returned. 2. Repeat this process with another weather forecast service. 3. If none are available, proceed monitoring without weather forecast data. 4. After 5 minutes try to reconnect.
Related requirements	FR-7

3.5.3 Subscribing to SMS Service

Number	UC-3
Description	Citizens can subscribe to the SMS service, so when a flood happens they will get a direct text message
Stakeholders and interests	**Citizens**: Citizens want to be warned as soon as possible.
Primary actor	Citizen
Scope	Warning part of the system
Level	User goal
Precondition	Citizen has a mobile phone and is not subscribed to the SMS service
Main success scenario	1. Citizen sends a text message to our SMS service 2. The SMS service receives the text message 3. The SMS service sends the phone number to the SFM system 4. The SFM system stores the phone number in the database 5. A text message is sent back to the citizen with confirmation by the SMS service
Postcondition	Citizen is subscribed to the SMS service
Alternatives	2a. The text message is not received The use-case ends
Related requirements	FR-16

3.5.4 Determining flood probability

Number	UC-4
Description	The central processing unit calculates the probability of a flood
Stakeholders and interests	• **Safety region**: The safety region wants to know when a flood warning is triggered • **Government**: The government would also like to know when a flood warning is triggered
Primary actor	System
Scope	Monitoring and warning part of the system
Level	Sub process
Precondition	The sensor data is available
Main success scenario	1. The central processing unit gets the latest sensor data from the database 2. The central processing unit gets the latest weather forecast data 3. The central processing unit calculates the probability of a flood 4. The central processing unit stores the probability value in the database 5. The central processing unit determines that a flood is imminent based on the probability value 6. A warning is send to the emergency services (they will warn the government) 7. A warning is send to the citizens
Postcondition	The flood probability is calculated and stored. If the probability exceeds a certain threshold, a warning is sent to the authorities and citizens
Alternatives	5a. The probability is not above the threshold The use-case ends
Related requirements	FR-8

3.5.5 Warn citizens in case of an imminent flood

Number	UC-5
Description	Citizens who are subscribed to the SMS service will be warned through text messages in case of an imminent flood
Stakeholders and interests	• **Citizens**: When they are subscribed, they want to be warned in case of an imminent flood
Primary actor	Citizen
Scope	Warning part of the system
Level	User goal
Precondition	There is an imminent flood and the citizen is subscribed to the SMS service
Main success scenario	1. The flood monitoring & detection unit sends a warning about an imminent flood to the warning unit 2. The warning unit queries the database for a list of phone numbers of subscribed citizens in the area 3. The monitoring unit sends the collected phone numbers to the SMS service 4. The SMS service sends a warning to all the collected phone numbers
Postcondition	The citizens who are subscribed received a warning
Alternatives	3a. 1. A message cannot be sent to the citizen 2. Wait a minute and resend 3. The use-case ends
Related requirements	FR-17

3.5.6 Warn safety region in case of an imminent flood

Number	UC-6
Description	The safety region needs to receive a warning about an imminent flood
Stakeholders and interests	• **Government**: The government wants to warn the citizens in case of a flood • **Safety regions**: The emergency services want to help the citizens in case of a flood
Primary actor	Safety region
Scope	Warning part of the system
Level	User goal
Precondition	There is an imminent flood
Main success scenario	1. The processing unit determines what area will be under water in case of a flood 2. The processing unit determines how many people will be affected by the imminent flood 3. The processing unit predicts how the flood will develop in the following period 4. The processing unit will create a map based on the current state and predictions 5. The processing unit sends the map to the government and emergency services
Postcondition	A map with current and predicted data is sent to the government and emergency authorities
Related requirements	FR-10, FR-11, FR-12, FR-13, FR-14

3.5.7 Third party accessing data through the systems API

Number	UC-7
Description	Third parties can use the API exposed by the flood monitoring system in third party applications (providing guidance to citizens)
Stakeholders and interests	• **Third parties**: Third parties want to access the data of the flood monitoring system to use in their applications • **Citizens**: Citizens want to receive guidance in case of a flood.
Primary actor	Third party
Scope	The API part of the system
Level	User goal
Precondition	The third party has access to the flood monitoring systems API
Main success scenario	1. The third party application connects to the API 2. The third party application sends a request for certain data to the API 3. The system retrieves the requested data from the database 4. The system sends the retrieved data to the third party application
Postcondition	The third party application has received the requested data
Related requirements	FR-28

3.5.8 Detecting a faulty sensor

Number	UC-8
Description	The system is able to detect when a sensor is not functioning properly
Stakeholders and interests	• **Product owner**: The product owner wants the system to be reliable and errors/broken sensors to be fixed
Primary actor	The system
Scope	The monitoring part of the system
Level	Sub process
Precondition	The sensor is not functioning properly
Main success scenario	1. The system receives the sensor data 2. The system compares the sensor data with other information, including data of nearby sensors and previous data of this sensor 3. The system detects that this reading is abnormal, but determines it cannot be caused by an (imminent) flood 4. The system ignores further readings from this sensor and reports the faulty sensor in the control panel
Postcondition	The faulty sensor is not used in future measurements and is reported in the control panel
Alternatives	3a. 1. The system cannot determine the abnormal reading is not caused by a flood 2. The system keeps using this sensors data, until it can determine that the readings are not caused by a flood, or it determines that it is caused by a flood (in which case it will issue warnings, see UC-4)
Related requirements	FR-18, FR-20, FR-21, FR-22, FR-23

3.5.9 Maintenance employee checks system state

Number	UC-9
Description	A maintainer of the system regularly checks the state of the system in the control panel to see if there are errors or sensors that need maintenance.
Stakeholders and interests	• **Product owner**: The product owner wants the system to be reliable and errors/broken sensors to be fixed
Primary actor	Maintainer
Scope	The maintenance part of the system
Level	User goal
Precondition	
Main success scenario	1. The maintainer uses his login credentials to get access to the control panel 2. The maintainer navigates to the errors/warning page 3. The maintainer checks on this page if there are problems with the system (errors/warning) 4. The maintainer takes necessary action to resolve any issues
Postcondition	The maintainer is aware of reported problems with the system
Related requirements	FR-19, FR-20, FR-21, FR-22, FR-23

3.6 Functional requirements

This section lists the functional requirements of the system.

Nr.	Prio	Description
FR-1	**Must**	The system is able to receive input from water level sensors. This information will be used to determine if there is an imminent flood.
FR-2	**Must**	The system is able to perform an analysis for the water level in the waterways based on the input from the water level sensors.
FR-3	**Must**	The system is able to receive input from the dike sensors.
FR-4	**Must**	The system is able to perform an analysis for the parameters of the dike sensors based on the input from the dike sensors.
FR-5	**Must**	The system can store the sensor data.
FR-6	**Must**	Sensor data, which has been previously stored, can be retrieved at a later moment.
FR-7	**Must**	The system retrieves weather forecasting data from weather forecasting services, which consists of predictions about the precipitation, wind data and tide information. This is used by the system to help in determining when a flood becomes imminent.
FR-8	**Must**	The system is able to detect when a flood is imminent by combining the retrieved sensor data and weather forecasting data.
FR-9	**Must**	The system retrieves geographic information, consisting of road data, terrain height data and demographic data (number of civilians living in affected area) from an external API.
FR-10	**Must**	The system computes the area affected by a flood, in zones of 5 by 5 km, by using the location data of the sensors and geographic information.
FR-11	**Must**	The system is able to perform an analysis, resulting in an estimated expected water level for areas which are affected by a flood, based on the water level sensor data, geographic data and weather forecast information.
FR-12	Should	The system estimates how the water level in the areas affected by the flood will develop for every hour, up to 12 hours in the future.
FR-13	Should	The system can compute the number of civilians living in the areas affected by the flood.
FR-14	**Must**	When a flood is imminent, the system sends a warning to the safety region, containing information about the flood: the area affected by the flood, the expected water level in those areas, how the water level will develop in the coming hours and the number of civilians living in the affected area.
FR-15	**Must**	The system can compute a safe area, not affected by the flood, where citizens can be evacuated to in case of an (imminent) flood.
FR-16	**Must**	Citizens are able to subscribe to flood warnings about imminent floods.
FR-17	**Must**	Citizens who are subscribed for flood warnings are warned about imminent floods by text message.
FR-18	**Must**	The system can detect a faulty sensor, either when the sensor raises an error or when the data from the sensor is inconsistent with other sensor data.
FR-19	**Must**	There is a control panel, where maintainers of the system have access to.
FR-20	**Must**	The system reports faulty sensors, so they can be viewed in the control panel.
FR-21	**Must**	Warnings of the system can be viewed in the control panel.
FR-22	**Must**	Errors of the system can be viewed in the control panel.
FR-23	**Must**	The readings of the sensors can be viewed in the control panel.
FR-24	**Must**	The system can make backups of its data (configuration data etc.).
FR-25	**Must**	The system can store created backups on a remote location.
FR-26	**Must**	The system can retrieve the backups it previously created.

FR-27	**Must**	The system can restore the backups it previously created after retrieving them.
FR-28	**Must**	The system exposes an API, allowing third parties to develop applications for guidance of the citizens during a flood.
FR-29	Could	The system is able to detect extreme weather phenomena, like storms etc.
FR-30	Should	The system processes and stores data collected using a UAV.

3.7 Commercial non functional requirements

In this section commercial non functional requirements are presented.

Nr.	Prio	Description
CNFR-1	**Must**	The system is affordable. The initial price of the system is lower than 95% of the competitors price in the same market.
CNFR-2	**Must**	The sensors have a good quality, so they do not have to be replaced often. The expected lifetime of the sensors should be at least three years.

3.8 Technical non-functional requirements

This section describes the technical aspects that are important to the system as requirements. These requirements determine various APIs and programs that the system will rely on.

3.8.1 Reliability

Reliability is an important non-functional requirement for the system, and a key-driver of the architecture as well.

Nr.	Prio	Description
REL-1	**Must**	Data from the sensors is sent via a TCP connection
REL-2	**Must**	The system must detect if a sensor supplies wrong measurements, which can be caused, e.g. by improper calibration or defects in the sensor.
REL-3	**Must**	The system must at no time fail to detect a flood when this flood becomes imminent (*false negative*).
REL-4	**Must**	The system must not detect a flood, when this flood is not there in reality (*false positive*), on average more than once per 5 years.

3.8.2 Availability

Nr.	Prio	Description
AVA-1	**Must**	The system must have an uptime of 99.7%. This effectively means, that the system should not be down for more than 2 hours per month. $AV = \frac{\text{MTTF}}{\text{MTTF+MTTR}} = \frac{\text{6 months}}{\text{6 months + 12 hours}} = \frac{\text{4380 hours}}{\text{4380+12 hours}} = 99.7\%$
AVA-2	**Must**	The system must not experience a period of downtime, spanning more than 12 hours. Within twelve hours of the system going offline, it should be back up again.

3.8.3 Resilience

The system needs to be resilient to recover from errors and mistakes without impacting the systems functionality.

Nr.	Prio	Description
RES-1	**Must**	The system recognizes failures within half an hour
RES-2	**Must**	The system recovers from failures without the Quality of Service or the functionality of the system being affected.
RES-3	**Must**	All system data must be backed up every 24 hours, so that in case of data loss, this data can be restored.
RES-4	**Must**	In case of a data loss, the data should be retrieved and restored from a backup within 2 hours.
RES-5	**Must**	Backup copies are stored in a secure location which is not in the same area as the system (50 km).

3.8.4 Performance

Nr.	Prio	Description
PERF-1	**Must**	Data is transmitted from and to the system with a minimum average speed of 10 megabits per second
PERF-2	**Must**	The data transmission between the sensors and the system is on average at least 10 megabits per second for each sensor.
PERF-3	**Must**	The time for the system to compute if there is a flood or not according to a critical level and the data received from the sensors is at most 5 minutes.
PERF-4	**Must**	If an imminent flood is detected, the warning text message to citizens arrives in 5 minutes.
PERF-5	**Must**	If an imminent flood is detected, the warning to the emergency room arrives within 1 minute.

3.8.5 Interoperability

The system has dependencies on several third-party systems and also allows third parties to retrieve information from it.

Nr.	Prio	Description
INTR-1	**Must**	The system pulls weather forecasts from at least two weather forecasting services.
INTR-2	**Must**	When the system detects a flood, it notifies the safety region through an API provided by them.
INTR-3	**Must**	The system sends out a SMS to all users who are subscribed to flood warnings using the mobile network.
INTR-4	**Must**	The systems is able to connect to different types of sensors.
INTR-5	**Must**	The system is able to retrieve geographical data from an API.
INTR-6	**Must**	The system exposes an API, allowing third parties to develop applications using the systems data.

3.8.6 Security

The security of the system is very relevant to its success. The system should be secure, because unauthorized access can have a big impact on society (when e.g. false flood warnings are triggered).

Nr.	Prio	Description
SEC-1	**Must**	Access to the system is restricted to users, which are authorized and authenticated using a password protected user account.

SEC-2	**Must**	All communication to, from and within the system are encrypted.
SEC-3	**Must**	User account information is hashed using bcrypt after being salted with 128 randomly generated characters.
SEC-4	**Must**	The system is protected by a firewall that at least scans at the application layer, while also scanning for and mitigating DDoS attacks.
SEC-5	**Must**	The system communicates with its sensors via a REST API that only allows for HTTPS connection.

3.8.7 Scalability

The system has to be designed in a way that it can expand over time. Not only should it span larger geographic areas, but more functionality will be added later as well.

Nr.	**Prio**	**Description**
SCALE-1	**Must**	The database and services of the system should run in a cloud environment where they can scale, when the systems resource usage increases.
SCALE-2	**Must**	The system is configurable to run in different areas and with different sensors.
SCALE-3	**Must**	The system maintains the performance requirements when the geographic area the system covers is expanded.

3.9 Evolution requirements

When establishing the project, architects of the system listed a certain number of requirements which describe the features of the system. However, due to environmental changes and changing stakeholder interests, for example, the requirements may evolve.

ER-1 : Adding of external input
Citizens can contribute to the guidance by giving extra information (for example if they identified a safe route near their location). This information can be checked by an operator (thanks to photograph by the UAVs for example).

Functionality can be added to the system in the future, that can take citizen feedback into account for the flood prediction.

ER-2 : New sensors The system is able to work with new sensors technologies, which are coming on the market over the years.

ER-3 : Improved algorithms for detecting a flood
New algorithms may become available, which have a better accuracy for the flood prediction described in FR-8. The part of the system with the flood prediction algorithm has to be modular, so that it can be replaced with an improved algorithm, once such an algorithm becomes available.

3.10 Risk assessment

The system is confronted by several risks which are determined and mitigated in this section. Taking those risks into account allows to avoid them or at least reduce their impact. The risk management involves the identification of the risks, their probability and potential impact or consequences.

The tables below explain the meaning of the definition for probability and consequence.

Probability	Likelihood of occurrence
High	0.65 - 1.00
Medium	0.35 - 0.65
Low	0.00 - 0.35

Severity	Explanation
Severe	A risk that can lead to loss of live or casualties.
Significant	A risk that can lead to damages, can delay the project more than 3 months or causes one of the high-level requirements not to be fulfilled.
Moderate	A risk that can lead to one of the high-level requirements not to be fulfilled to an acceptable level.
Minor	A risk that can lead to one of the high-level requirements not being fully fulfilled, but still fulfilled in an acceptable level.

3.10.1 Technical

T-RISK1	**The system does not detect a flood**
Probability of occurrence	Low
Consequences	Severe. There can be a loss of human lives and damages, loss of trust in the system by end-users.
Prevention	Make sure the number of sensors is sufficient and that they are in good state (as low failure rate as possible, when necessary repair or replace them). Perform regular checks of the sensors. Make sure faults in sensors are reported.
Reaction	Make changes in the algorithm for the flood detection, improve the sensors used or add more sensors.

T-RISK2	**The system sends warnings of a non-existing flood (false positive)**
Probability of occurrence	Low
Consequences	Significant. People can become more negligent to future messages and unneeded social disturbance can be caused.
Prevention	UAVs watching the area where the supposed flood is to confirm.
Reaction	Send a message as soon as the mistake is detected to tell the population/emergency center it was a false alert.

T-RISK3	**The system cannot send messages to the necessary people because the communication platform is also destroyed by the flood**
Probability of occurrence	Medium
Consequences	Severe. If the warning is not send, the area might not be evacuated timely. Potential loss of human lives, casualties and damages to property.
Prevention	
Reaction	Send the warning to the government using a different medium.

T-RISK4	**Hacker gets access to the system**
Probability of occurrence	Low
Consequences	Severe. The hacker may sent incorrect information deliberately during the flood. This can cause unneeded evacuation, but in the case of a flood also loss of human lives. The system is not reliable anymore.
Prevention	Change password and hash codes every three months. Hire specialists in the security field to audit the security system on a regular basis (penetration testing).
Reaction	Update the security system / change it. Find a new algorithm for the creation of password and hash codes.

3.10.2 Business

B-RISK1	**Wrong estimation of the budget**
Probability of occurrence	Medium
Consequences	Significant. The final product does not have the features expected.
Prevention	The team needs an accountant or at least someone taking care of the follow-up of the money. Make sure there are regular evaluations to keep track of the money flow.
Reaction	Remove some requirements or features of the product, or change the hardware components used.

B-RISK2	**The money invested in the fabrication and achievement of the product/system is not covered by the sales (shortfall/deficit)**
Probability of occurrence	Medium
Consequences	Moderate. Stopping the sale
Prevention	The team needs an accountant or at least someone taking care of the follow-up of the money.
Reaction	Adding more features to the product in order to make it more competitive in the market.

B-RISK3	**Third-party developers do not build third-party applications using the systems API**
Probability of occurrence	Medium
Consequences	Moderate. Without third-party applications, the citizens do not receive guidance in case of a flood
Prevention	Make sure the API exposes all features which are relevant to develop a third-party application for guidance.
Reaction	Promote the use of the API by third-party developers using, for example, a contest.

B-RISK4	**Sensor becomes unavailable (is not sold anymore)**
Probability of occurrence	Medium
Consequences	Moderate. Sensors will fail and need replacing over time, if new sensors are not available anymore, this becomes impossible.
Prevention	Choose a sensor that is not too old and is expected to be available for at least the next 5 years.
Reaction	Use a different sensor and modify the system so it can operate with this sensor.

3.10.3 Schedule

S-RISK1	**The project is not finished at the deadline**
Probability of occurrence	Low
Consequences	Significant. Pressure for all the team members, loss of credibility regarding the customers, selling a product with less features than expected.
Prevention	SRA , Schedule Risk analysis : Estimation of the duration of the project by its manager (with the use of probability and statistics) . Meeting for the team members every week to keep track of the timing and take decisions according to the deadline.
Reaction	Remove some requirements or features in order to finish the project as soon as possible.

4 Analysis

This chapter describes the analysis of the system. It lists the assumptions that are made about the system and its environment. Next, an overview of the high-level design decisions is given.

4.1 Assumptions

There are several assumptions made about the system and its environment:

1. The safety region / government has means to alert citizens in an area to evacuate.
2. The safety region is informed about our system and will alert citizens if needed when our system alerts the safety region.
3. People who are subscribed for flood warnings/guidance have a mobile phone that can receive text messages.
4. Sensors can be placed in the water ways and dikes in locations where they can provide representative measurements.
5. Information is available to the system with regards to the population density and terrain height in different areas.
6. Supplying information about the areas affected by the flood in a resolution of 5 by 5 km (as described in FR-10) is sufficient for the government/safety region to make decisions regarding alerting and evacuating citizens.
7. It is possible to predict the expected water level up to 12 hours in the future.
8. A weather forecasting API is available that provides precipitation and wind data.
9. The safety region provides an API, which can be used by the system to warn the safety region.
10. The information about the flood warning can be displayed in the safety region using an API provided by them.

4.2 High-level Design Decisions

This section discusses the high-level design decisions.

Each decision is explained in a table. The arguments section of the table lists for each alternative a score for every important quality attribute. A higher score means a more favourable result. For example, a high score for costs means a low cost for the system.

Name	**Operating system**
Decision	**DEC-1**
Status	**New**
Problem/Issue	The warning system software for the natural disasters need a platform to work on.
Decision	The warning system will use Linux as a platform. Based on Unix, Linux is a free platform that has proven itself and is used by many servers. It's open source meaning that everyone can check out how it works.
Alternatives	*Windows* Operating system is a closed platform developed by one of the biggest tech companies who provide a big development environment with it. *OpenBSD* A Unix-based system that is famous for it's proactive security and runs most of the Linux applications. However, some software packages aren't certified to run on OpenBSD, but are for Linux.
Arguments	

	Reliability	Resilience	Performance	Interopertability	Security	Scalability	Cost	**Score**
Weight	1	1	1	1	1	1	1	
Linux	4	4	4	4	4	4	5	29
Windows	3	2	3	2	3	3	1	17
OpenBSD	5	4	4	3	5	4	3	28

Table 4.1: Decision – Operating system

Name	**Connectivity of the sensors**
Decision	**DEC-2**
Status	**New**
Problem/Issue	The sensors need to deliver their data to the system and are located outdoors with at least 100m distance between each other.
Decision	The sensors will send their data to the system using mobile broadband. Using cellphone towers to communicate with the system.
Alternatives	*Landline* Connecting the sensors to the telephone network and use that network to communicate with the server. *Satellite* Set up a satellite connection between the sensors and the system. *Direct lines* Connection the sensors to the system by drawing a cable from all sensors to the system.

Arguments

	Reliability	Resilience	Performance	Interopertability	Security	Scalability	Cost	**Score**
Weight	1	1	1	1	1	1	1	
Mobile broadband	4	4	4	4	2	5	4	27
Landline	2	2	3	4	3	3	5	22
Satellite	3	1	2	4	4	4	3	21
Direct lines	2	1	5	2	5	1	1	17

Table 4.2: Decision – Connectivity of the sensors

Name	**Type of water level sensor**
Decision	**DEC-3**
Status	**New**
Problem/Issue	To measure the water level in the water ways and along the coast, a sensor is needed to measure the water level.
Decision	The system will use pressure sensors to measure the water level. Pressure sensors are submerged at a fixed level in the water body. By measuring the pressure on the sensor of the water above it, these types of sensors are able to determine the water level.
Alternatives	*Float-operated sensor* These types of sensors are mechanical and have a floating element which can move up and down with the water level. The floating element is protected in a 'stilling well' and therefore, the risk of damage is low. *Non-contact sensor* These types of sensors use (ultra)sonic waves to determine the water level. Sediment in the water can cause issues with the measurements. *Bubbler sensors* Bubbler sensors can measure the water level by measuring the pressure needed to force an air bubble through a submerged tube. Bubbler sensors have good resistance against damage from floods and debris.
Arguments	For this decision the interoperability and security have a weight of 0, because those depend on the specific sensor used and not on the type of sensor. Scalability was not taken into account because this depends mostly on the costs of the type of sensor.

	Reliability	Resilience	Performance	Interopertability	Security	Scalability	Cost	**Score**
Weight	3	2	1	0	0	0	3	
Pressure sensor	3	3	3	-	-	-	4	30
Float-operated sensor	3	4	3	-	-	-	3	29
Non-contact sensor	2	4	3	-	-	-	2	23
Bubbler sensors	4	5	2	-	-	-	1	29

Table 4.3: Decision – Type of water level sensors

Name	**Server**
Decision	**DEC-4**
Status	**New**
Problem/Issue	To connect all sensors and to process all the data, we need computing power and storage space.
Decision	We will use our own server park. When managing our own server park, we have more technical freedom. Especially when our system will scale up, it can be profitable to maintain our own server park instead of a cloud solution.
Alternatives	*Cloud computing* There are different cloud computing providers in the world. An advantage of cloud computing is the degree of scalability. However, when storing data at a cloud provider, we can never be sure if we are the only ones with access to our data.
Arguments	For this decision we will not take resilience into account. Resilience can also be seen as reliability and interoperability.

	Reliability	Resilience	Performance	Interopertability	Security	Scalability	Cost	**Score**
Weight	3	-	2	2	3	2	2	
Cloud computing	4	-	4	3	3	5	3	51
Own server park	3	-	4	5	4	3	4	53

Table 4.4: Decision – Server park

5 System architecture

This chapter describes the general system architecture of the SFM. The first section describes the different kinds of input and outputs the system needs. The verification section verifies the decisions made in the system architecture. The last section goes into more detail and explains the internal architecture.

5.1 System context

The system context is a fundamental artifact in the software architecture of a system. Developing the system context view is important, because this view is used as a mechanism to trace back to the business context, and downstream to the functional and operational architecture.

5.1.1 Diagram

The system context diagram is outlined in Figure 5.1. This diagram represents the different users and external parties that are involved in the SFM. In this diagram, the system is represented as a black box. In section 5.3 a more detailed view of the system is given.

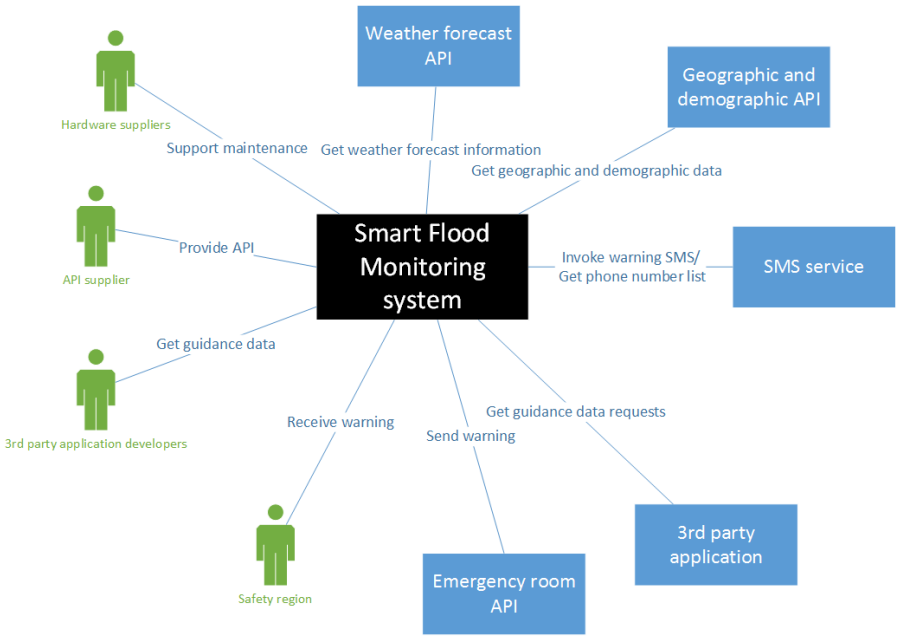


Figure 5.1: System context diagram

Figure 5.1 only depicts external and internal connections. Internal user is not depicted in the diagram. However, internal users, such as system administrator and maintenance user, are able to connect to the system, which is authorized and authenticated using a password.

5.1.2 Users and Roles

Safety region The safety region has the responsibility to protect citizens when a dangerous situation occurs. They receive a warning when there is an imminent flood. When the warning is received, the safety region can warn citizens and emergency services. The safety region needs information about time to evacuate, location and severity of the imminent flood.

Third party application developers This user will build an application to provide guidance when a flood is imminent. They use the data provided by the API of the SFM. They are allowed to get all relevant data from the system.

Hardware suppliers This user will provide different kinds of hardware to the system and could act as a repairman when hardware is failing and can't be repaired right away.

5.1.3 External Systems

Weather Forecast API The system will utilize weather forecast services from third party sources. To make this input reliable, the system uses multiple weather forecast providers. To predict floods correctly the system will need: rain data, temperature data, tides data, air pressure data and wind data.

Geographic and demographic API To predict how floods will evolve over time, the system needs geographical data. There are multiple parameters that are needed by the system: ground height and waterways. To calculate the impact of an imminent flood on society, the system also needs to know how many people live in the affected area. This data is retrieved through the demographic API.

SMS service The SMS system will communicate with the system and the mobile phones of the citizens. When a citizen wants to subscribe to the SMS service, the citizen can send a text message to the SMS provider containing the postal code and the house number. This data is then send to our system, which stores the information. When a flood is imminent and a warning needs to be issued, the system determines which citizens should receive a warning based on their address. The list of phonenumbers that is composed is then used to send a message to all citizens who are in the affected area. To do this, the SFM uses the API of the same SMS provider.

Third party application To guide citizens to a safe area in case of an imminent flood, we rely on third party applications. These applications get data from our API and deliver an app for citizens that provides guidance to a safe area.

Emergency room API In case of an imminent flood we need to warn the safety region. This is done by invoking the emergency room API. In this way we send a message to the emergency room, which in their turn distributes this warning to the safety region.

5.1.4 Channels and Information Flows

This subsection elaborates channels and information flows from and to the SFM. All connections are encrypted to ensure the security of the system.

$SFM \Leftrightarrow Weather\ forecast\ API$	
Description	The system gets rain, temperature, tides, airpressure and wind data.
Connection	Wired, internet
Protocol	TCP/IP
Transaction occurence	Real time

$SFM \Leftrightarrow Geographic\ and\ demographic\ API$	
Description	The system gets ground height data, waterway data and the amount of people who live in the affected area.
Connection	Wired, internet
Protocol	TCP/IP
Transaction occurence	Real time

$SFM \Leftrightarrow SMS\ Service$	
Description	The SMS service receives subscriptions and sends warning messages to the right phone numbers when it gets warned by the system.
Connection	Wired, internet
Protocol	TCP/IP
Transaction occurence	In case of an imminent flood

$SFM \Leftrightarrow Third\ party\ applications$	
Description	The system will provide data to the third party applications through the API.
Connection	Wired, internet
Protocol	TCP/IP
Transaction occurence	Real time

$SFM \Leftrightarrow Emergency\ room\ API$	
Description	The system warns the the safety region through the emergency room API.
Connection	Wired, internet
Protocol	TCP/IP
Transaction occurence	In case of an imminent flood

5.1.5 Alternatives

Weather forecast data

The system will use the openweathermap API. This is an API which can deliver both historical and forecast data. There are other options to get the current and forecast weather data, one of these options is forecast.io. We use this weather API if the openweathermap is unavailable.

Geographic and demographic data

The system will use the Nationaal Georegister (NGR) API to get the latest geographic data. To get the latest demographic data the system will use the open data from the Centraal Bureau van de Statistiek (CBS). This provider delivers both actual and historical data. It would also have been possible to download the data and import it just one time. However, the data needs to be up to date. This means the data needs to be updated once in a while, the easiest way to do this is by invoking the API.

SMS Service

To warn the citizens in case of an imminent flood, we use the external CM Telecom SMS service. Citizens can subscribe to this service and receive a warning. Another option is that the system uses its own SMS service. The advantage of this is that we keep control of the process of warning citizens. However, it would be expensive to implement such a service. Besides this, outsourcing is more scalable since we don't have to cope with problems of sending text messages to other countries.

Another way of informing citizens would be to send a whatsapp message. However, all mobile phones are able to receive text messages, while not all mobile phones have internet access to receive whatsapp messages.

Emergency room

The emergency room will be warned by invoking their API. This decission was made to make sure all data will be received correctly. By warning the emergency room through telephone it would be possible the employee

could accidently forget or change some of the information. If this would be implemented the SFM would need an employee 24/7, which is expensive.

Another option is to send an email in case of an imminent flood. However, in this case it wouldn't be possible to continously update the information in the dashboard of the emergency service. This could also be achieved by continously sending mails, but this would take more time to open and interpret the emails.

5.2 Verification

The feasibility of the Smart Monitoring is verified in the following section.

5.2.1 Availability

To determine the availability of the system we determine the availability of the separate components. Let:
MTBF: Mean Time Between Failure
MTTR: Mean Time To Repair
The availability of the complete system is given as x
$x = \frac{MTBF}{MTBF+MTTR}$

Component	MTBF	MTTR	Availability
Internet connection	3 years	1 day	0.99908
Storage	5 years	1 day	0.99945
API	3 years	1 day	0.99908

The system availability can be calculated based on the table above. This is done by multiplying all seperate availabilities. The result of this is 0.9976. This meets the requirement of the system to be available 99.7% of the time.

5.2.2 Cost

The external services are not all free. This means this is another expense. To use the openweathermap the system needs the service of $470 a month. The back-up for the openweathermap is forecast.io, this gives 1000 calls per day free, after that each call costs $0.0001. The geographic and demographic data is free in the Netherlands. In case of an expansion to other countries, other third party services may be looked into.

5.3 Elaborated Model

In figure 5.1 the system architecture model is shown. This figure is an elaborated version of the system context figure in chapter 5.1. The arrows at the end of the lines represent data flows, going in that direction. First of all the system retrieves data from external API's. The data is retrieved by the data collection system and send to the database. The same applies for the UAV and sensor data. Besides this data flow, there is also a data flow from the data collection system to the UAV. In case the system detects that it needs UAV data, the data collection system sends commands to the UAV. The data collection system also retrieves the phone numbers and addresses of the citizens who have subscribed to the SMS service. After all data is collected the algorithm component will determine the flood probability based on the known data. If it detects that a flood is imminent, a warning message is sent to the warning system. This warning part then warns the emergency room by invoking the API. It also sends a text message to all the citizens that are in the affected area. To provide data to the third party applications the system holds an API. This API can be used by third party developers to get relevant data from the database.

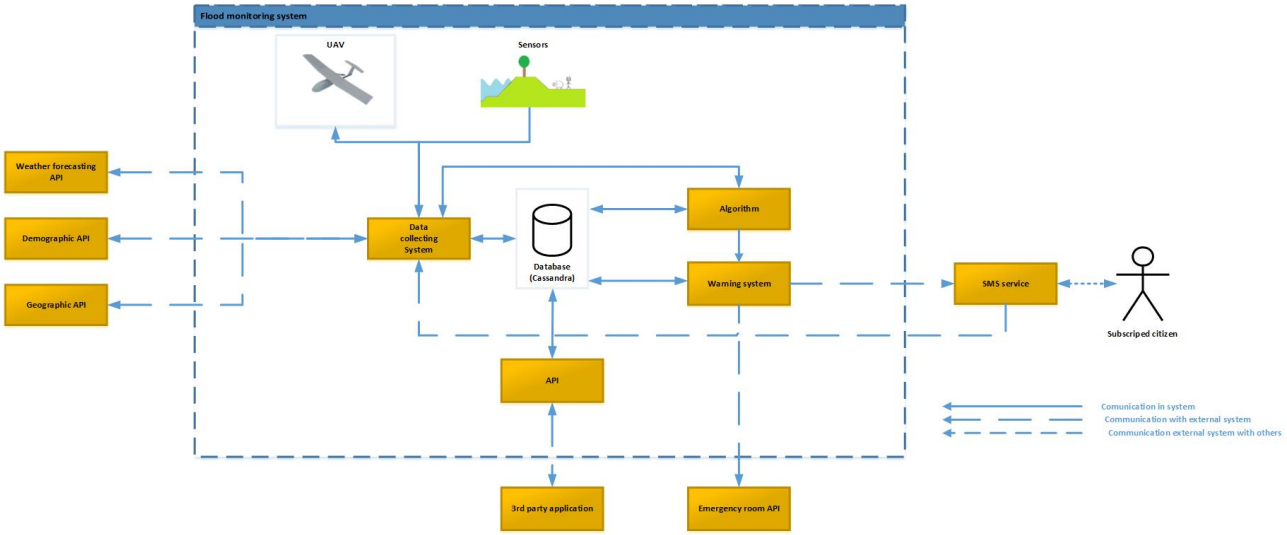


Figure 5.2: Elaborated system context diagram

6 Hardware Architecture

This section describes the hardware architecture of SFM. The description will be more high-level along with explanations about the hardware platform and the application interfaces between each components of the system. The rest of this chapter is organized as follows; First section, section 6.1, presents an overview of the hardware implemented in this system depicted in big schema. Decisions made in this system are detailed in section 6.2 with tables. Lastly, the hardware is described in section 6.3.

6.1 Hardware Overview

The SFM hardware components can be categorized into three main components: sensing part, data storing part, and analytics part. The data flow starts from wired and wireless sensors located across the Netherlands that collect information for monitoring. UAV will also fly to gather additional information if needed. The overview of the hardware and its application interfaces are depicted in Figure 6.1 below.

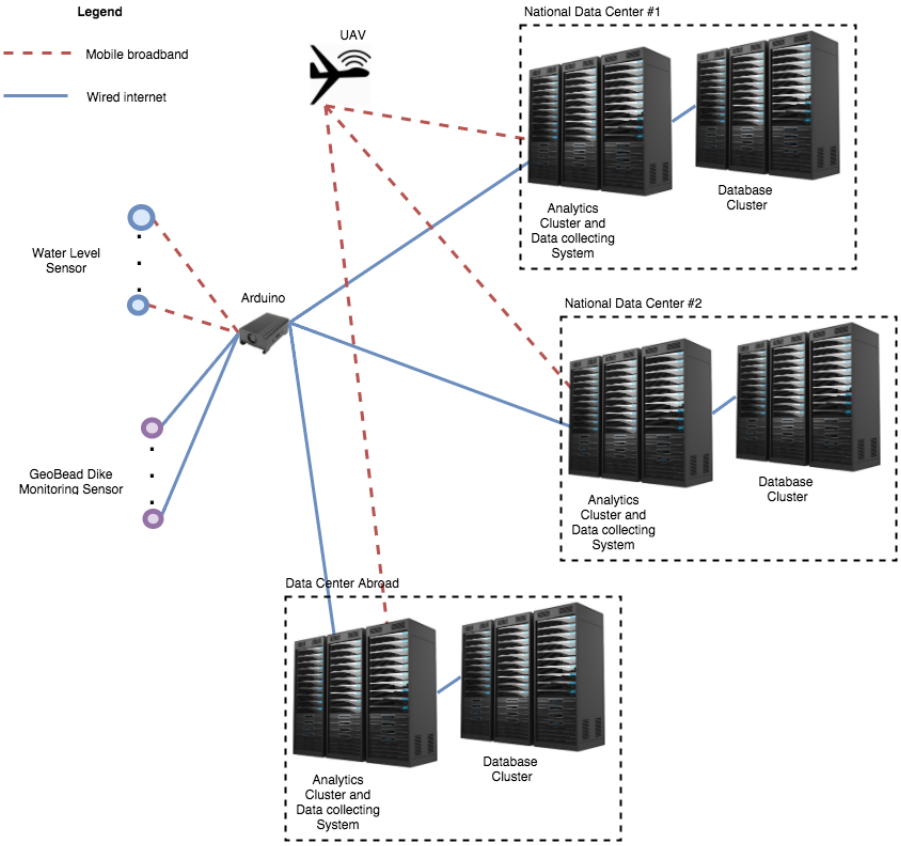


Figure 6.1: Schematic overview of the hardware architecture of Smart Monitoring

The SFM will utilize wired and wireless sensors to monitor water ways and dikes. Wired sensors will be used for dikes monitoring and wireless sensor will be used to monitor water level. Sensors are not directly connected to the main clusters, instead, it will be connected to Arduino and then will be sent periodically using wired Internet connection through telecoms provider's line. An Arduino will be handling several sensors at once.

UAVs will also fly to check reported faulty sensors if needed. UAVs will also be used to take required pictures for further analytics or to examine some portion of the system which is hard or impossible for a personnel to access.

All incoming data will be handled by Data Collecting System in the data centers. This system is also responsible to check any incorrect data input or any faulty sensors. The next part of the hardware is the cluster for carrying

out analysis. This will be a collection of servers that are coordinated using clusters. SFM will use another cluster to store important data. This cluster will run Elasticsearch database on top if it.

The last part of the hardware architecture is the third party data gathering cluster. The third party data gathering cluster is responsible for collecting weather forecast and demographic information of the Netherlands. This cluster is also part of the main analytics clusters.

The SFM will have several data centers to keep the system reliable running as reliability and availability is SFM's key drivers. Two of them will be inside the Netherlands and the other one will be located abroad. In this way, the system will be running correctly in case both national data centers go down. The data centers are also protected by a firewall that at least scans at the application layer, while also scanning for and mitigating DDoS attacks.

6.2 Hardware Design Decisions

This section defines decisions made regarding the hardware selection. Tables will be used to make our justification in regard to hardware selection more crystal clear.

Name	**Choice of dike sensor**
Decision	**HW-1**
Status	**Approved**
Problem/Issue	The system needs a reliable sensor system to measure condition of dikes.
Decision	The system will implement GeoBeads MEMS Sensor in dikes.
Alternatives	*GeoBeads* The GeoBead is a compact sensor, which can measure the pore pressure, temperature and local tilt in dikes. A unit costs about 350 dollar[22]. *Piezometers* Piezometers measure the pore water pressure in the dikes. This information can be used to measure the stability of the dike. A piezometer costs about 200 dollar[22]. *Volt meters* Volt meters can be used to measure the streaming potential in the dike, which are an indicator of its stability[21]. These sensors are approximately 50 dollars per unit. Materials in the dike can decrease the accuracy of this measurement technique.
Arguments	

	Reliability	Resilience	Interoperability	Cost	**Score**	
Weights	3	2	3	2	2	
GeoBeads	5	4	4	3	2	45
Piezometer	4	2	3	2	3	35
Volt meters	1	5	3	2	5	36

Table 6.1: Decision – Choice of Sensors

Name	**Connectivity of the dike sensor**
Decision	**HW-2**
Status	**New**
Problem/Issue	The dike sensors have to be connected to the internet in some way, so they can send their data to the central server.
Decision	The sensors will be connected by a wire.
Alternatives	*Wired* A wired cable connects the dike sensor. This cable can also be used to supply the sensor with electricity. *ZigBee* ZigBee is an open protocol for personal area networks. It uses little power and is therefore a good choice for devices equipped with a battery. *ISM radio band* The ISM radio bands can be used for industrial, scientific and medical purposes.
Arguments	ZigBee is not an option, since the sensors are embedded in the soil of the dikes, and the frequency it uses, decreases too much in strength when traveling through the dike[24]. Research has been done by van der Gees and Kok [24] to determine if it is feasible to use the ISM radio band to communicate from within the dike. They concluded that, while it is possible to communicate through the dike using this band, the distance is limited and the rate of error is relatively high. While a wire is not ideal in the sense that it will have to connect all the sensors, it seems to be the best option for the sensor in the dikes. It has the additional benefit that it can also supply the sensors with electricity.

	Reliability	Resilience	Performance	Interoperability	Security	Scalability	Cost	**Score**
Weights	3	2	3	2	2	2	2	
Wired	5	4	3	3	2	2	2	50
ZigBee	4	2	2	2	2	4	3	42
ISM Radio band	1	5	2	2	2	3	5	43

Table 6.2: Decision – Connectivity of the dike sensor

<table>
<tr><td>Name</td><td>Data Gathering from Sensors (using Arduino)</td></tr>
<tr><td>Decision</td><td>HW-3</td></tr>
<tr><td>Status</td><td>New</td></tr>
<tr><td>Problem/Issue</td><td>There must be single board computer that handles connection from sensors to Data Collecting System.</td></tr>
<tr><td>Decision</td><td>Arduino Uno Ethernet will bridge between sensors and Data Collecting System.</td></tr>
<tr><td>Alternatives</td><td>Arduino Uno
A low cost but powerful single board computer that can be programmed as needed. Arduino Uno has high extendability because it can be equipped with Arduino shield that will expand its capabilities. The minus point of this type is it does not have built in Ethernet jack.
Arduino Uno Ethernet
This is basically the same as Arduino Uno. However, it has built in Ethernet jack.
Arduino Mega
This type of Arduino has more flash storage and SRAM. However, the physical size of this device is also significantly bigger.</td></tr>
<tr><td>Arguments</td><td>The most suitable Arduino for the SMF is the Arduino Uno Ehternet because it has built-in Ethernet jack. Thus, the remaining expansion port will remain free for future evolution purpose. This way, the energy consumption of Arduino will also be low because it does not have any extension shield. We assume that Arduino will be powered by power landline connection of telecom provider.
<table>
<tr><th></th><th>Reliability</th><th>Resilience</th><th>Performance</th><th>Interoperability</th><th>Security</th><th>Scalability</th><th>Cost</th><th>Score</th></tr>
<tr><td>Weights</td><td>3</td><td>2</td><td>3</td><td>2</td><td>2</td><td>2</td><td>2</td><td></td></tr>
<tr><td>Arduino Uno</td><td>4</td><td>3</td><td>3</td><td>3</td><td>2</td><td>3</td><td>5</td><td>53</td></tr>
<tr><td>Arduino Uno Ethernet</td><td>4</td><td>3</td><td>3</td><td>5</td><td>2</td><td>3</td><td>4</td><td>55</td></tr>
<tr><td>Arduino Mega</td><td>4</td><td>3</td><td>3</td><td>3</td><td>2</td><td>3</td><td>3</td><td>49</td></tr>
</table></td></tr>
</table>

Table 6.3: Decision – Arduino selection

Name	**UAVs**
Decision	**HW-4**
Status	**Approved**
Problem/Issue	The system uses UAV's in order to get more data during a flood and to create a map of the flooded area.
Alternatives	*Skyjib 8 Titanium* Flight mode : GPS, computer based, Time of flight : 12-25 min , Speed : 12m/s. *DJI Phantom 3* Flight mode : GPS, comupter based, Time of flight : 15 minutes , Speed : 10 m/s . *Skyjib Super 6 Ti-QR* Flight mode : GPS, comupter based, Time of flight : 10 minutes , Speed : 10 m/s . Can only support GroPro.
Arguments	Getting high quality data quickly : photograph and 306 deg videos . Produce accurate and high resolution surveys . Cost effective. View to hard-to-reach and dangerous(where the man cannot go) areas which ensures safety because it does not require personnel to enter potentially hazardous environments. Establishment of Elevation Models to assess water flow direction and accumulation,.

Table 6.4: Decision – UAVs

Name	**Analytic cluster selection**
Decision	**HW-5**
Status	**Approved**
Problem/Issue	SFM needs a reliable computers to do the analytical processing.
Decision	SFM will use clustered Dell PowerEdge R530 to act as the main analytic cluster and to provide API to the actors.
Alternatives	*HP ProLiant DL360 Gen9 Base* This server rack has 16GB of memory and 2.4GHz of processor speed. As other server computer, this machine utilizes Intel Xeon E5 2600v3. This server is suitable for high dense computing, however the price is not so suitable for this kind of specification. It does not have LCD screen that will help technician to look the current status of the server. *Lenovo System x3550 M4 7914* This server rack has only 8GB of memory. However, the processor is a bit faster, it runs on 2.6GHz. As other server computer, this machine also utilizes Intel XEON E5-2600. The price is a little bit lower than the others but the memory limitation makes it not so valuable. It has LCD screen that will help technician to look the current status of the server. *Dell PowerEdge R530* This 2U server rack has 16GB of memory and 2.4GHz of processor speed. This machine utilizes Intel Xeon E5-2620V3 with 15MB of cache. This server is suitable for high dense computing. It has LCD screen that will help technician to look the current status of the server.
Arguments	

	Reliability	Performance	Interoperability	Security	Scalability	Cost	**Score**
Weights 3	2	3	2	2	2	2	
Dell PowerEdge R530 5	3	5	4	4	4	5	70
Lenovo System x3550 M4 7914 4	3	4	4	3	4	4	60
HP ProLiant DL360 Gen9 Base 5	3	5	4	3	4	3	64

Table 6.5: Decision – Analytic cluster selection

Name	**Database cluster selection**
Decision	**HW-6**
Status	**Approved**
Problem/Issue	The system needs reliable computers to store the data.
Decision	SFM will utilize Synology RackStation RS814RP to store the data.
Alternatives	*Synology RackStation RS814RP* This storage machine has the fastest connection among the others. This machine will run at SATA with 6 Gbps connection. *70BJ NAS-server* This machine form factor is 1U which is suitable for saving space. However, the connection speed is limited to 3 Gbps. *Thecus N8810U-G NAS-server* This machine also runs in 3Gbps connection. However, the form factor is 2U which makes this machine takes more space in the rack.

Arguments

	Reliability	Performance	Interoperability	Security	Scalability	Cost	**Score**
Weights 3	2	3	2	2	2	2	
Synology RackStation RS814RP 5	2	4	4	4	4	4	63
70BJ NAS-server 4	2	3	4	4	4	3	55
Thecus N8810U-G NAS-server 4	2	3	4	4	4	3	55

Table 6.6: Decision – Choice of storage machine.

Name	**High Performance Switch selection**
Decision	**HW-7**
Status	**Approved**
Problem/Issue	SFM cluster needs a powerful high performance switch to connect each cluster.
Decision	SFM will use Cisco Catalyst 2960S-24TS-L Switch.
Alternatives	*Linksys LGS552P Switch* This switch has 52 ports available for connection, which is very good for scalability. However the performance is not as good as Cisco catalyst switch series. *HP 1820-48G Switch* This switch has lesser available ports than Linksys switch. It has 48 ports available. This switch is not very configurable which makes this not so suitable for high performance switching. *Cisco Catalyst 2960S-24TS-L Switch* This switch has the lowest number of port available, 24 ports. However, Cisco Catalyst is very configurable and has a very good security.
Arguments	

	Reliability	Performance	Interoperability	Security	Scalability	Cost	**Score**
Weights 3	2	3	2	2	2	2	
Cisco Catalyst 2960S-24TS-L Switch 5	2	5	4	5	3	4	66
Linksys LGS552P Switch 4	2	4	4	3	4	5	60
HP 1820-48G Switch 3	2	4	4	3	4	5	57

Table 6.7: Decision – Choice of high performance switch

6.3 Hardware Description

This section gives an outline of the hardware implemented in this system. This section also elaborates on hardware decisions.

6.3.1 Sensor Components

Roughly 17.000 kilometers of dikes protect the Netherlands against flooding[3]. Of this, about 3.500 kilometers are primary dikes[14]. These are dikes protecting against the water of the sea, the big rivers (Rijn, Maas, IJssel), the IJsselmeer and the Markermeer.

The hardware architecture of the system is composed of two sensor components. The first sensor component are sensors installed in the dikes to measure and detect potential instability of the dikes. The second sensing component consists of water level sensors, which are spread along water ways in the country, to measure the water level.

Dike sensors

To measure the stability of the dikes, the GeoBeads dike monitoring sensor will be installed in the dikes. When these sensors are placed in the dike with a spacing of 3 meters, they will provide optimal measurements[22]. The GeoBead will measure the water pressure, temperature, inclination and acceleration.

A GeoBead sensor consists of several modules, loosely connected by cable, which form a chain of sensors. The GeoBeads will be installed in a pre-drilled vertical hole, or where this is not possible, horizontally. Three sensors are installed per cross-section and a cross-section is installed every 100 m.

The GeoBeads cannot send their data directly to the central server. Therefore, at each cross-section, there will be a module (consisting of an Arduino in a water-proof casing), which is responsible for processing and sending the data to the central server using cable internet.

The GeoBeads are connected by communication cable, and a cable for the power supply.

Water level sensors

The water level sensors are placed next to the dikes and in the water ways. The water level sensors are placed more scattered than the dike sensors. The water level sensors are also connected to an Arduino for processing and sending the data. The water level sensors are connected to power by cable.

The water level sensors are accommodated with a connectivity chip, which allows it to use the mobile broadband to connect to the central server.

6.3.2 UAV's

Goals : Flood monitoring and flood damage assessment UAVs provide a way for us to assess more accurately the impacts of climate change and flooding, The system will use UAV's in order to get more accurate data , and high resolution photographs and videos in order to map the flooded area.

High resolution elevation models for accurate predictions of flood extent thresholds and flow accumulation analysis. Elevation models can also be used to accurately measure changes to the physical land, such as landslides.

GPS, which allows te system to input coordinates in advance to map out the perfect flight plan. The UAV's won't fly everytime , they will be released during the risky periods of flood.

6.3.3 Database Cluster and Data Collection

In the previous chapter, SFM will use Elasticsearch database as the database platform. However, Elasticsearch requires computers to run its environment. SFM will use clusters of computer to manage database system and to store our data. Furthermore, user account information is hashed using bcrypt after being salted with 128 randomly generated characters to increase security. The cluster will also be accessible by the main analytics part as the data will come and go through the main analytics part. The logical schematic of the database cluster is depicted in Figure 6.2

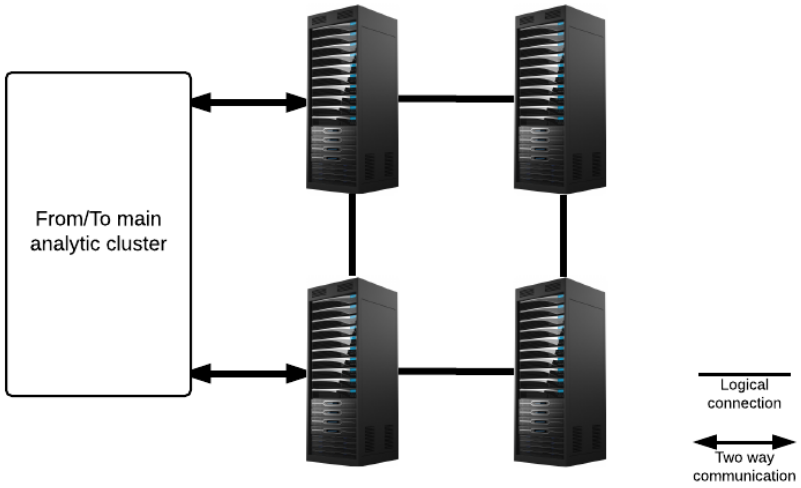


Figure 6.2: Logical schematic of database cluster of SFM

As can be seen in Figure 6.2, SFM will use four database racks to have redundancy in the system. The server is connected as a ring, which is the common way to setup database server. The database cluster By using this form of architecture, SFM will be more reliable and fault tolerant. There will also be two physical connection to the main analytic cluster to make this system more fault tolerant in terms of connection. SFM database cluster will use the same server, Dell PowerEdge R530, for controlling the SATA storage machine.

6.3.4 Analytics Components

Analytics component will be the main brain of SFM. The intelligent algorithm will run on this machine. This components are also responsible for checking faulty sensors by analyzing incoming sensor data. Thus, there is a big dependency to this components. To increase availability and reliability, SFM will have six server racks to do the processing as depicted in Figure 6.3.

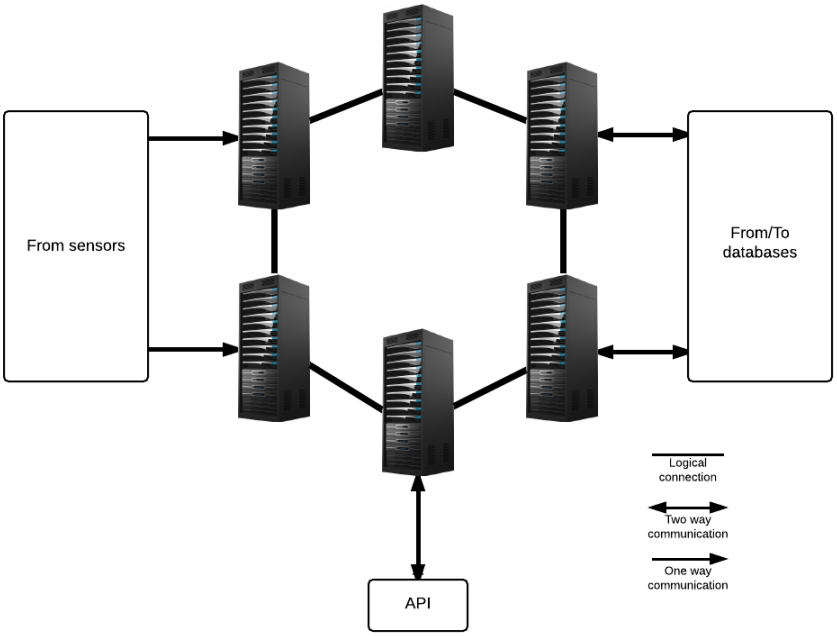


Figure 6.3: Logical schematic of analytic cluster of SFM

The analytics components will also be the hub for the main connection of SFM. Thus, this system also need a high performance switch to accomplish this purpose. As have been mentioned before, this system will use Cisco Catalyst 2960S-24TS-L Switch.

The analytics components will use Dell PowerEdge R530 as server and it will be mounted on a server rack. The detailed hardware specifications of Dell PowerEdge R530 is listed in Table 6.8.

Dell PowerEdge R530 Specification	
Item Description	Dell PowerEdge R530 - E5-2620V3 Xeon 2.4 GHz - 16 GB - 1 TB
Type	Server - rack-mountable
Height (Rack Units)	2U
Processor	1 x Intel Xeon E5-2620V3 / 2.4 GHz (3.2 GHz) (6-core)
Processor Main Features	Intel Turbo Boost Technology 2
Cache Memory	15 MB
Cache per processor	15 MB
RAM	16 GB (installed) / 384 GB (max.) - DDR4 SDRAM - 2133 MHz
Storage Controller	RAID (SATA 6Gb / s) (Dell PERC H330)
Optical Storage	DVD burner
Graphics Controller	Matrox G200
Video Memory	16 MB
Network	GigE
Dimensions	(WxDxH)48.24 cm x 64.6 cm x 8.68 cm
Weight	2.14 kg

Table 6.8: Hardware specification of Dell PowerEdge R530

7 Software Architecture

This chapter describes the software architecture of the SFM. This will be described in these four sections: software architecture design (attribute-driven design), architectural view, components, and software design decision.

7.1 Software architecture design

This section states software architecture design of the system with attribute-driven design method. As mentioned in [1], the Attribute-Driven Design (ADD) method is a systematic step-by-step method for designing the software architecture of a software-intensive system. It is an approach to defining software architectures by basing the design process on the architecture's quality attribute requirements. It follows a recursive decomposition process where, at each stage in the decomposition, tactics and architectural patterns are chosen to satisfy a set of quality attribute scenarios.

7.2 Architectural view

The following section will elaborate the views of the system into 4+1 Model: logical view, implementation view, process view, and deployment view.

7.2.1 Scenarios 

This section describes the scenarios that are most likely to happen in our system. Scenarios are the interactions from our system with the different actors. First a textual step by step scenario is given, after that a diagram is shown with the scenario diagram.

Flood scenario

1. The sensors monitor the water level
2. The algorithm requests weather forecast data
3. The weather forecast API returns the requested data
4. The algorithm sends a request to the UAV to validate results or provide extra images
5. The UAV provides the requested validation or extra images
6. The algorithm sends a warning to the warning part
7. The warning part requests the phone numbers from all subscribed citizens in the affected area
8. The database returns a list containing the corresponding phone numbers
9. The warning part invokes the emergency room API
10. The warning part invokes the SMS API, which sends a SMS to all phonenumbers on the list

The different steps that are taken in the flood scenario are also represented in figure 7.1. This figure represents the logical connections that are made when a scenario is executed. This means that in reality a connection that is shown in the figure may not be made directly in reality. For example, the sensor data doesn't go directly from the sensors to the algorithm, there are some steps in between.

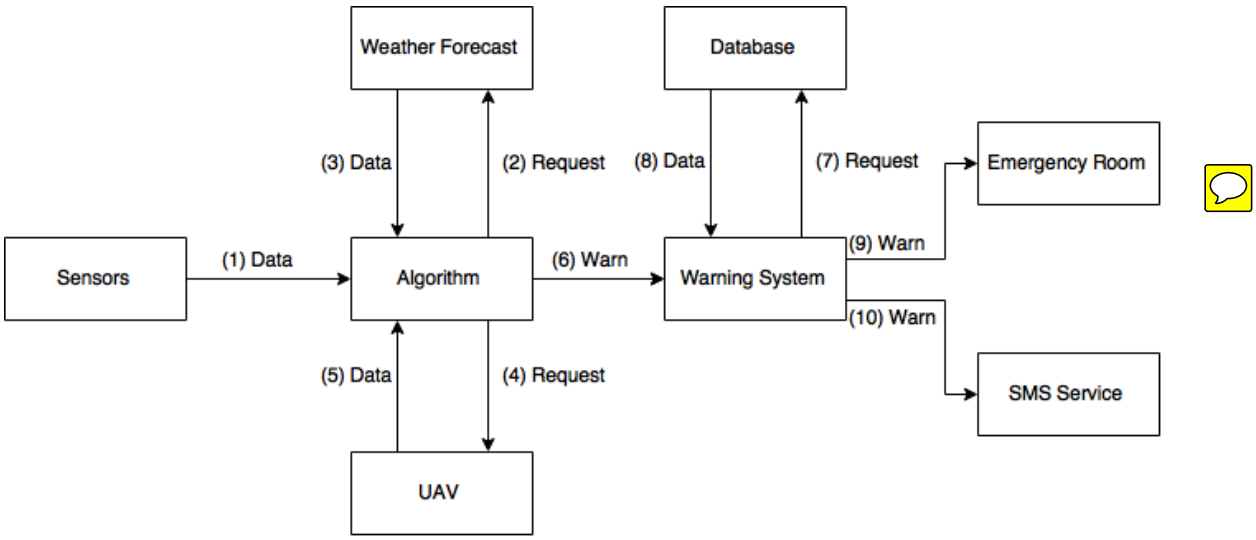


Figure 7.1: Scenario for a flood

Guidance scenario

1. A citizen receives a warning text message
2. A citizen requests guidance from a third party application to reach a safe area
3. The third party application requests the current flood information from the third party API
4. The third party API returns the requested data
5. The third party application returns a route to a safe area

When a flood is imminent or happening, a citizen can decide to request guidance to a safe area. If a citizen is subscribed to the SMS service, the citizen will get a warning text message in case of an imminent flood. Then the citizen can use a third party application that provides routes to safety that are as safe as possible.

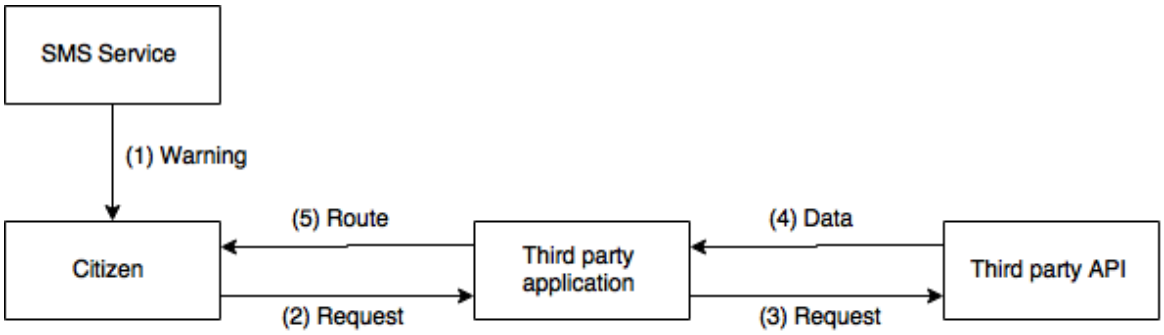


Figure 7.2: Scenario for guidance

7.2.2 Logical View

The logical view shows the structural elements, key abstractions and mechanisms that are needed to realize the SFM. First an overview of the different components is provided. After that the components are decomposed and more details of the different layers are provided.

Layer pattern

The software architecture of SMF is separated into three different layers. This layering is done according to the layering Martin Fowler proposes for enterprise applications [20, 13]. Layering the software provides a more flexible architecture that allows modifications of layers having to alter the other layers.
The different software layers of SMF are listed below. Each of these layers will be discussed in more detail after the figure.

Service layer This layer provides services to external systems and coordinates the incomming calls to the domain layer.

Domain layer The domain layer holds the core components for processing data and warning users.

Data source layer This layer stores all relevant data that is needed or produced by the system.

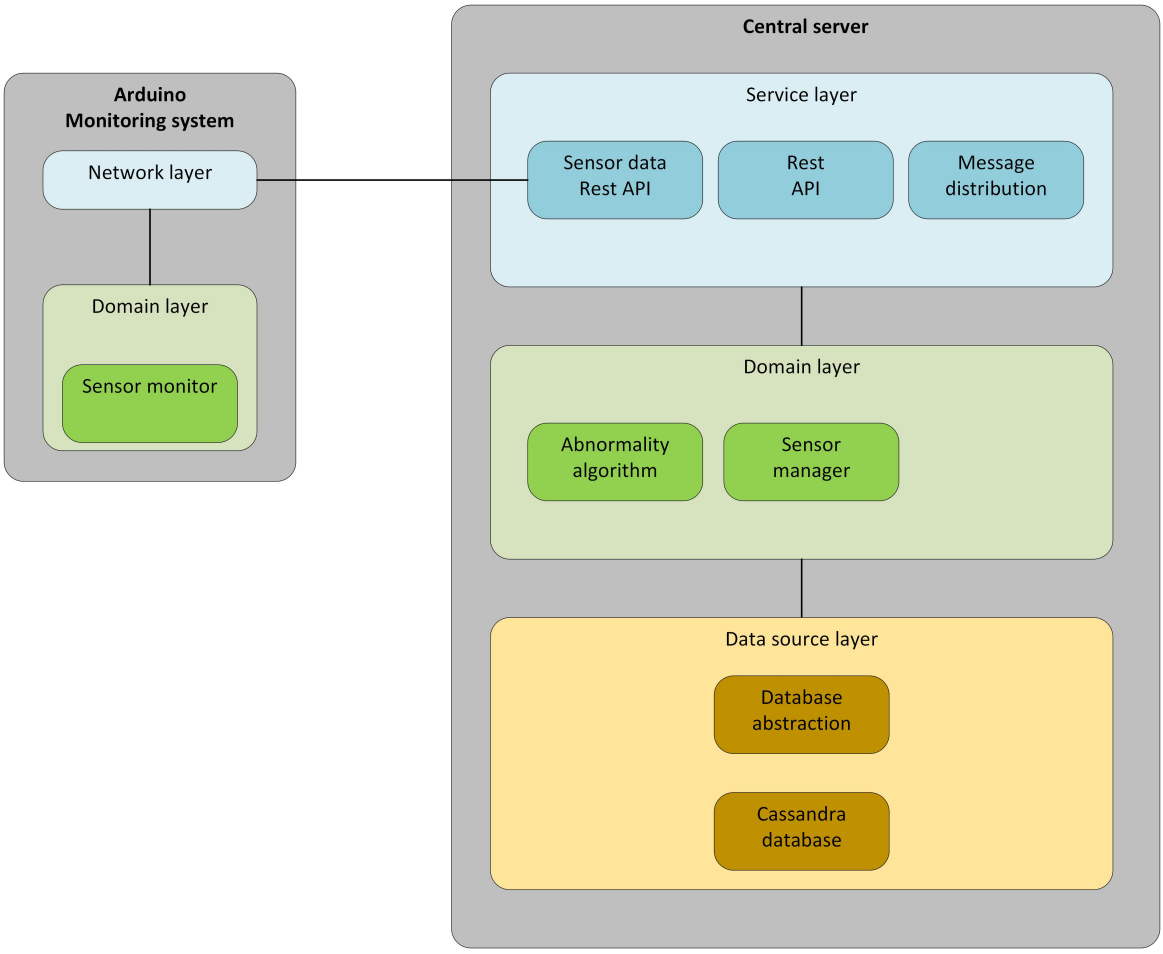


Figure 7.3: Layers of the software

Service layer

The Service layer pattern [20] is used as domain logic pattern. In this pattern, the service layer provides a set of operations that can be executed in the application. The service layer serves and coordinates the calls to these provides operation. The application logic in the domain layer is then separated from how it is called and how it should respond.
The main functionality of this layer consists of:

- Providing a REST server the arduino monitoring units can send their raw sensor data to
- Providing a REST server the third parties can use to receive flood data
- Distribute warning messages

The class diagram for this layer is shown in figure 7.4

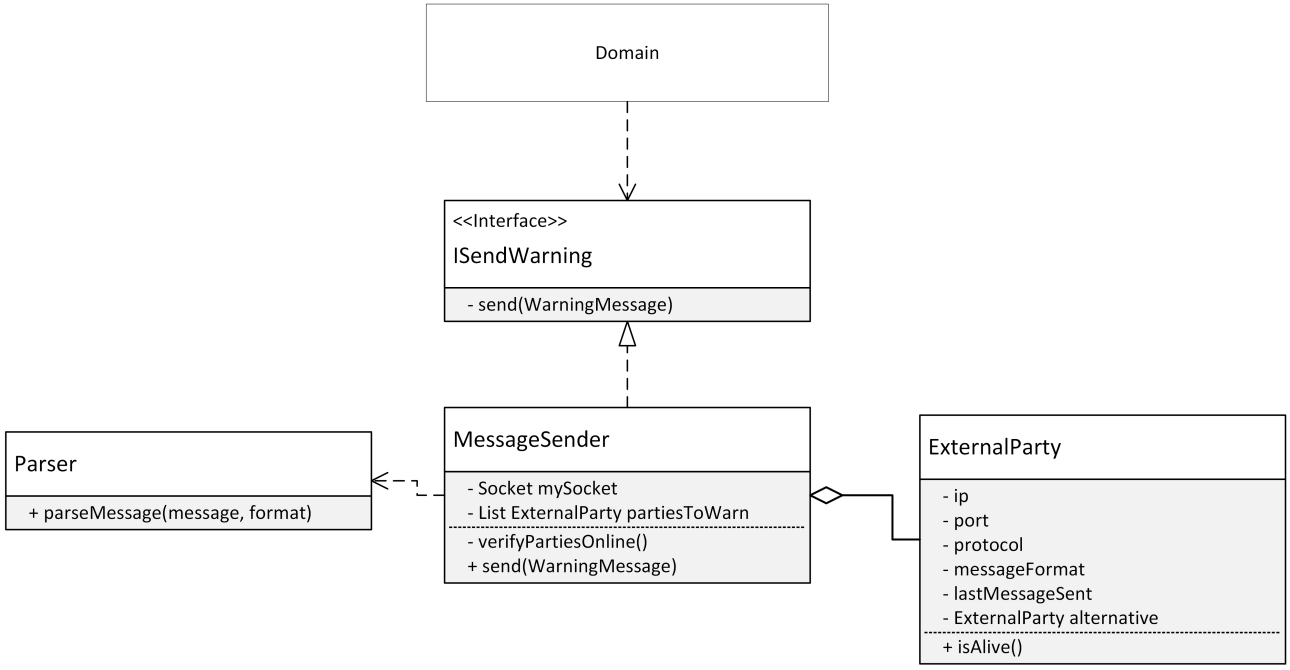


Figure 7.4: Class diagram of service layer

In figure 7.5 a sequence diagram of sending the sensor data can be seen. As elaborated in decision DEC-5, the sensors will push their data to the server. The sensor has two running threads: one reading the measurements and the other for sending the data every 60 seconds.

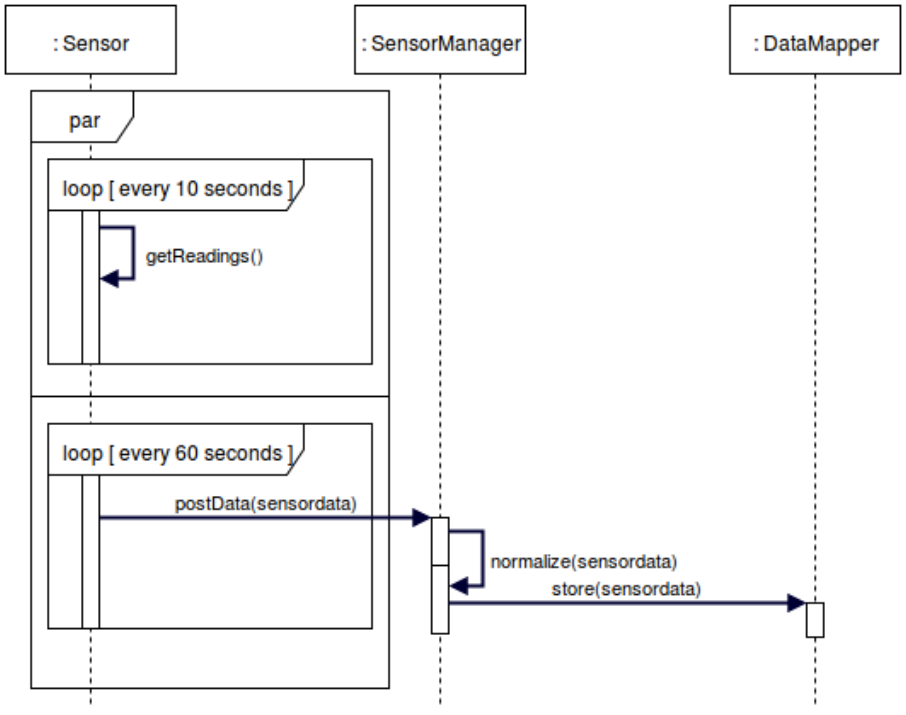


Figure 7.5: A sequence diagram of sending the sensor data

URL

`/monitor/{monitorid}/sensorvalues`

REQUEST HEADERS

```
Content-Type: application/json
Authorization: <<Hash>>
```

REQUEST BODY

```
[
        {
                sensor: "<<sensorid>>",
                value: "<<value>>"
        },
        {
                sensor: "<<sensorid2>>",
                value: "<<value2>>"
        },
        ...
]
```

The arduino system only pushes the data to the server. This push model reduces the load of the arduino systems because there are no incoming requests. This way, the arduino also doesn't have to store or manage any data, which also means the arduino doesn't need security measurements to secure this data.
The server does however need to verify that the messages it receives from the arduino monitoring systems are in fact send by our arduino systems. Otherwise a hacker could send sensor messages too, which would jeopardize the entire flood warning system.The downside to having the arduino systems push the data, is that data will be received by the server was not needed or wanted. Defect sensors data will still be send.

The message security is done by using a hash based message authentication code (HMAC) using SHA-256 as the hashing algorithm. Upon receiving the messages, the service layer verifies the message before further processing the message.
The sequence of actions for obtaining information from SFM is shown below in figure **??**.

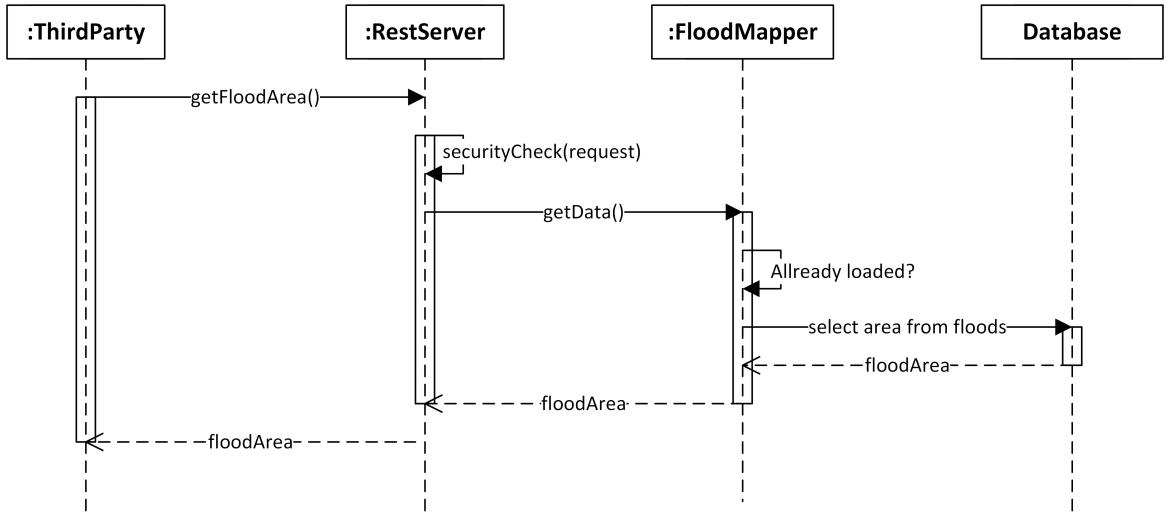


Figure 7.6: Sequence diagram of the client pushing the sensor data

The API calls the service layer receives that request information, don't need to be authenticated. These calls, however, do need to be limited by the service layer to not be able to delete or modify information. The information the API provides is defined in the server layer and the API does not return any information other then that.
The third party sends a http GET request to the REST server of SMF. Upon receiving this request, the REST server checks if the request is secure. This logic is located in the service layer. When the request is secure, the REST server calls the mapper object in order to get the requested data. The mapper object first checks the identity map if the data is already loaded. This will reduce the amount of queries executed on the database. If this identity map doesn't hold the data, the mapper sends a query to the database. The data returned from the database is then returned to the REST server, who sends the data to the third party.

Distribute warnings In figure 7.7 a sequence diagram of sending warnings to the emergency room and citizens can be seen. The sequence diagram contains a time constraint of 10 seconds and 5 minutes, for warning the emergency room and all citizens respectively.

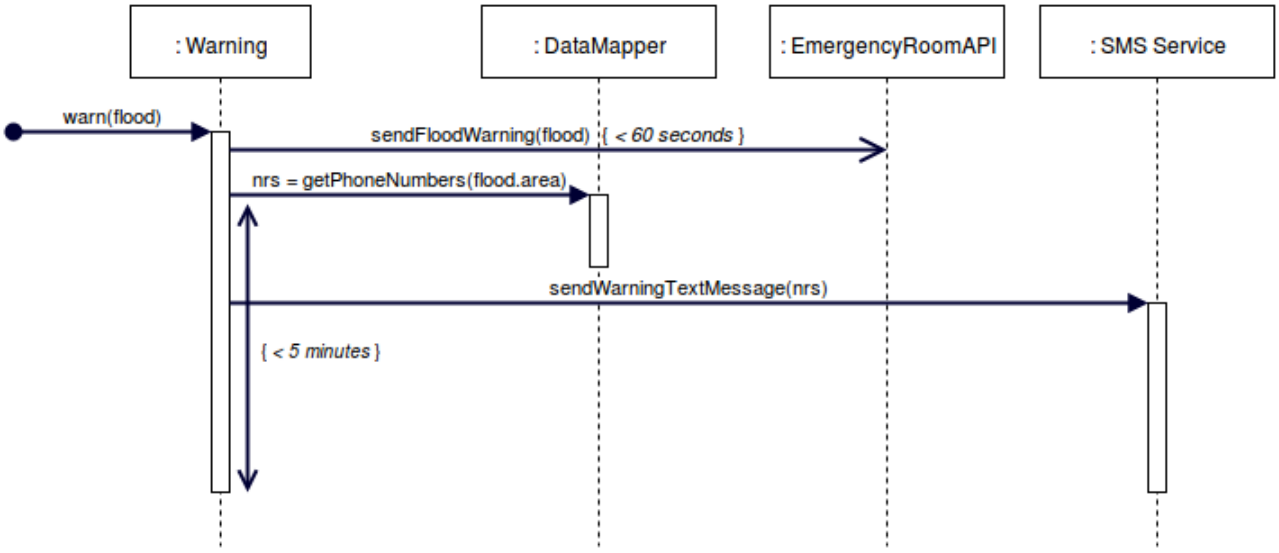


Figure 7.7: A sequence diagram of sending warnings to citizens and emergency room

Domain layer

The domain layer contains all the logic of the system. This includes the logic for the algorithms used to decide if a warning should be issued. And it includes the logic for managing the sensor data.
The sensor manager checks if the sensor is still working correctly.If the sensor is working correctly, the sensor data gets stored and processed.
The class diagram for handling incoming REST server calls is shown in figure 7.8.

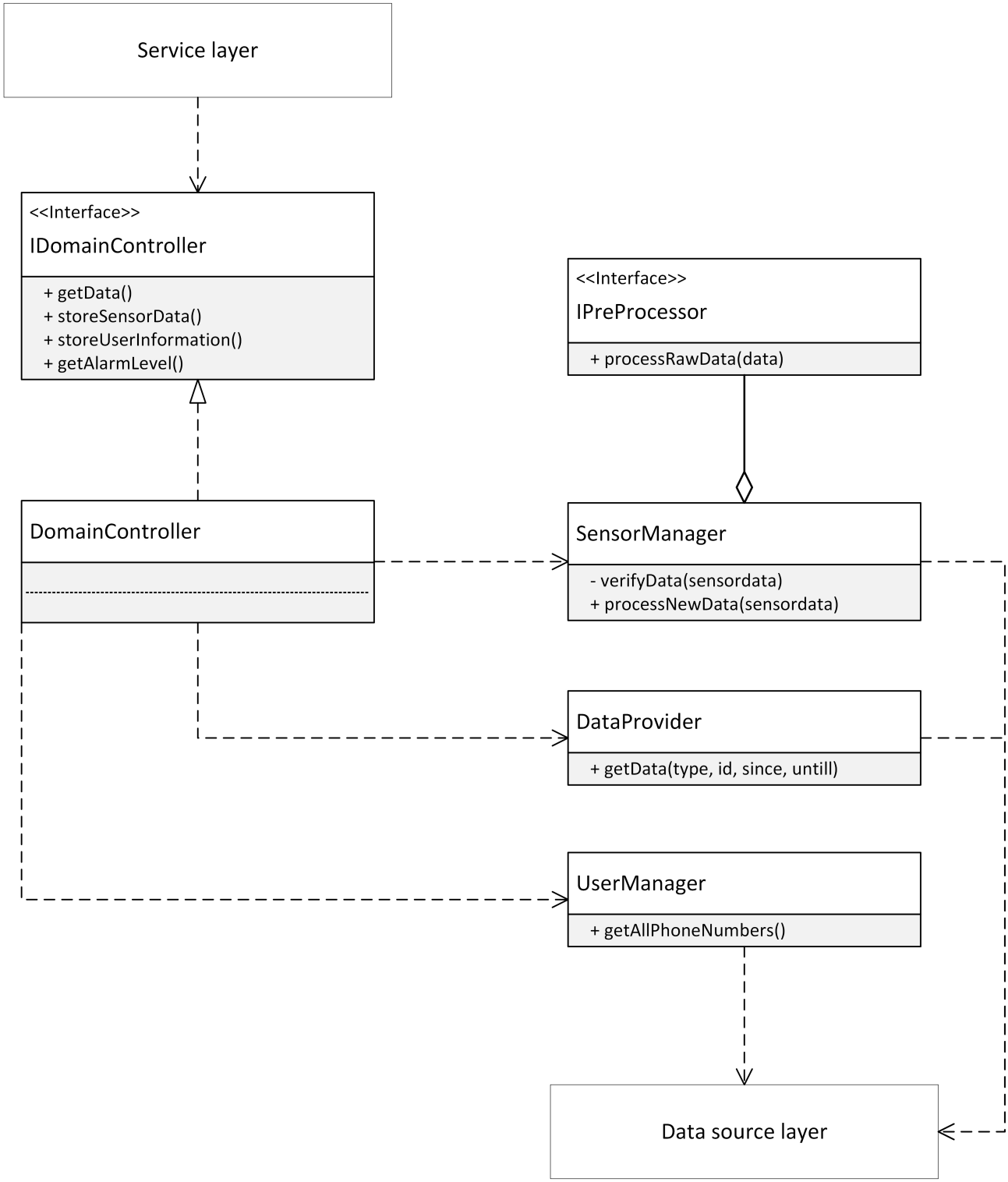


Figure 7.8: Class diagram of the domain layer concerning REST calls

The classdiagramm of the algorithm processing the data is shown below in figure 7.9

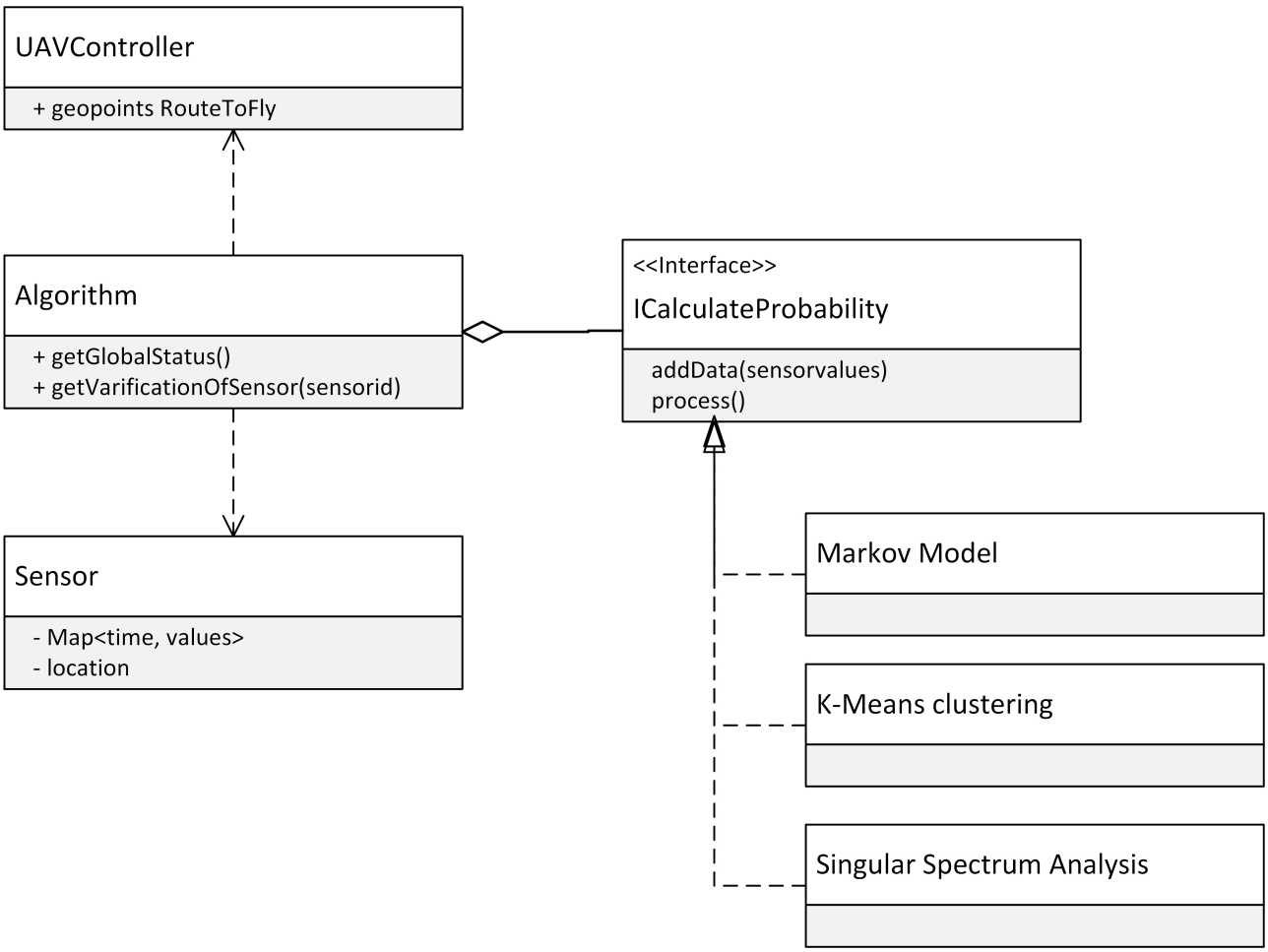


Figure 7.9: Class diagram of flood prediction in the domain layer

First the incoming sensor data is getting pre processed. This is done by using:

- Smoothing the L1 norm using the Huber function
- Hodrick–Prescott filter

The the L1 norm smoothing fluctuates more than the Hodrick-Prescott function. This way, the long term and short term differences can be analyzed better.

After the pre processing, the sensor data gets stored and will be used in several algorithms to calculate a flood probability. There are several algorithms being used, these are:

- Hidden Markov model with k-means clustering
- Singular Spectrum Analysis (SSA) for locations with tides
- Limit checking

Limit checking is just checking if the value of the sensor is not above a certain limit. A sudden extreme value will be detected using limit checking. If there are more sensors giving extreme values, the domain layer will verify this sudden change and might then issue the service layer to send a warning message.
SSA splits the range of values into blocks and then checks for abnormalities within that block.
The k-means clustering algorithm is used to detect abnormalities in the sensor data coming from a certain range dike. This way all the sensors should measure about the same values. In order to check for abnormalities, the k-means clustering can detect when a sensor value is out of the cluster.

How strongly correlated two types of data are will be calculated using Longitudinal Data Analysis.

Data source layer

The data mapper [20] is used to completely separate and ignore the database in the domain model. The domain object only uses an interface to a "find" function that returns the requested object. This could be either from the database or from the internal memory.
How the database is structured will be discussed in the implementation view section.

7.2.3 Implementation view

The implementation view, also known as the development view, describes the system from the programmer's perspective. This includes a description of the components of the system and how the system is packaged.

7.2.3.1 Components

The system consists of a set of components tat all interact with each other. In figure 7.10 below, these different components are shown. Each component can provide interfaces and uses sockets to connect to interfaces of other components. This way, the implementation of each component can independently be replaced and deployed.

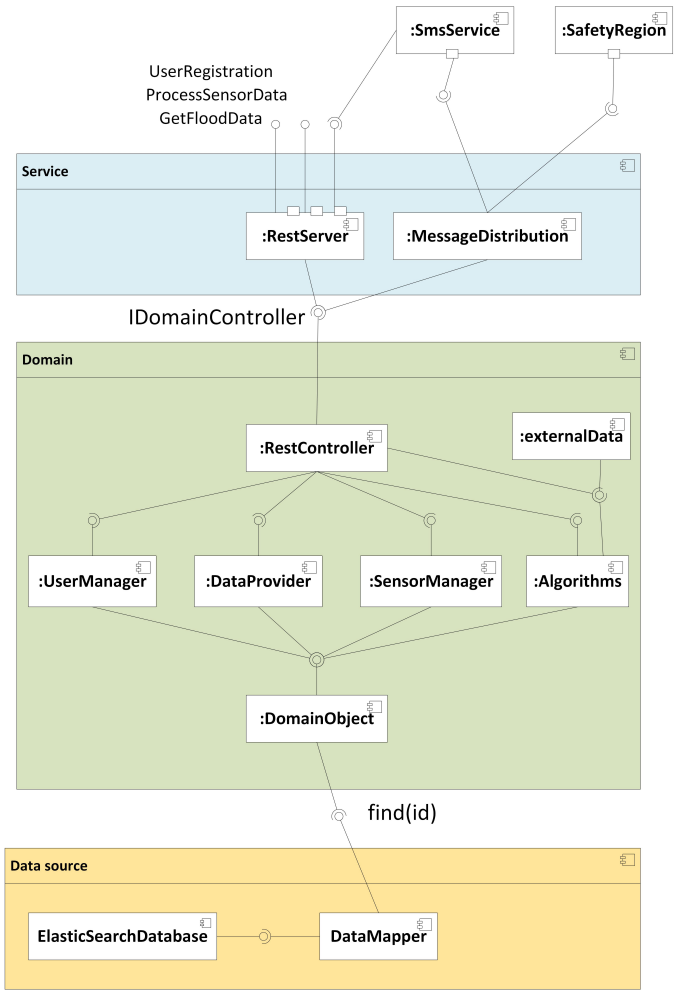


Figure 7.10: Component diagram

There components are placed in three layers. The data source layer, domain layer and service source layer.

In the data source layer the data is stored and the data mapper. It consist of two components.

- Database: This is the database that holds all the stored data of the system.
- DataMapper: This component is responsible for reading from the database and adding new data.

In the domain layer the business rules and logic is placed. The following components are placed:

- UserData: Responsible for gathering the user data and normalizing
- SensorData: Responsible for gathering sensor data and normalizing
- FloodData: Responsible for gathering flood data and normalizing
- UAVData: Responsible for gathering UAV data and normalizing
- UserManager: Responsible for maintaining the phone numbers and other information of the registered users.
- DataProvider: Responsible for obtaining the data the REST server requests.
- SensorManager: Component that is used for installing, update and delete sensors and retrieving data from the sensors.
- Algorithms: Component used for algorithms that calculated the flood probability.
- RESTController: Component to enable RESTful support.

In the Service layer the services are included used by external actors. The following components are included:

- RESTServer: Server which communicates with external software
- MessageDistribution: Component responsible for distributing warning messages to external parties.

Outside the layer are external software components:

- SMSService: Component to let citizen subscribe for warning texts and receive warning texts.
- SafetyRegion: Component to which the system communicates to in order to warn the Safety region.

7.2.3.2 Packages

The software of the system is divided into several packages. These packages and their relations can be seen in figure 7.11. In this diagram, «use» depicts the use of an interface exposed by a software package, «access» depicts a private import of (parts of) another package, while «import» means a public import of (parts of) another package.

The diagram contains software packages of the system itself, as well as software packages provided by third parties. The software packages provided by third parties are drawn outside of the 'Smart Flood Monitoring System'-box.

The 'Third Party API' is the software package which exposes a REST-API, which can be used by third parties to query data from the system. The 'External Data' is the software package which uses APIs from other systems to query data, like weather and geographic data.

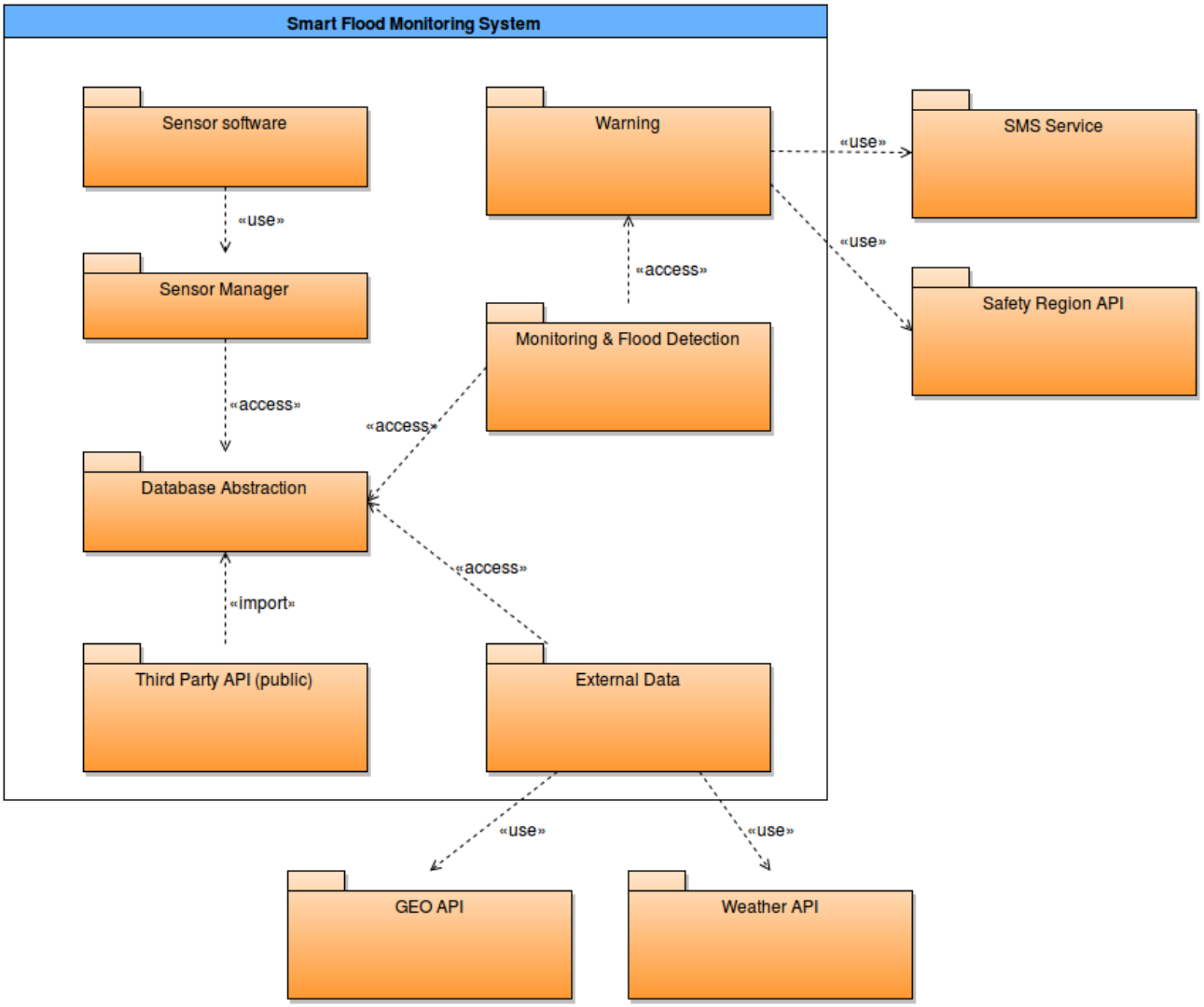


Figure 7.11: Package diagram

The following packages are in the system.

- Sensor Software: This software package is used for retrieving data from the sensors. It uses the sensor manager.
- Sensor Manager: Responsible for installing sensors and configure the sensors.
- Data Abstraction: Holds all data that is stored in the system.

- Third Party API: Package that enables third party software too access data from the system database.
- External Data: This package retrieves data from the GEO API, Weather API and Demographic API.
- Monitoring & Flood detection: Access the data from database and calculates if a flood is imminent.
- Warning: When a flood is imminent the warning package is used to send out a warning.

The following external packages are used:

- SMS Service: This package is used to let citizen subscribe to the warning services and send out warnings.
- Safety region API: When a flood is imminent a warning is send by the system to the safety region API package.

7.2.4 Process View

This view mainly discuss about runtime, concurrency, communication, and synchronization of the process running in the system.

The program flow and business logic of the system are captured in this section with the aid of activity and sequence diagrams.

Flood monitoring

The activity diagram in figure 7.12 shows the flow of the flood monitoring process.

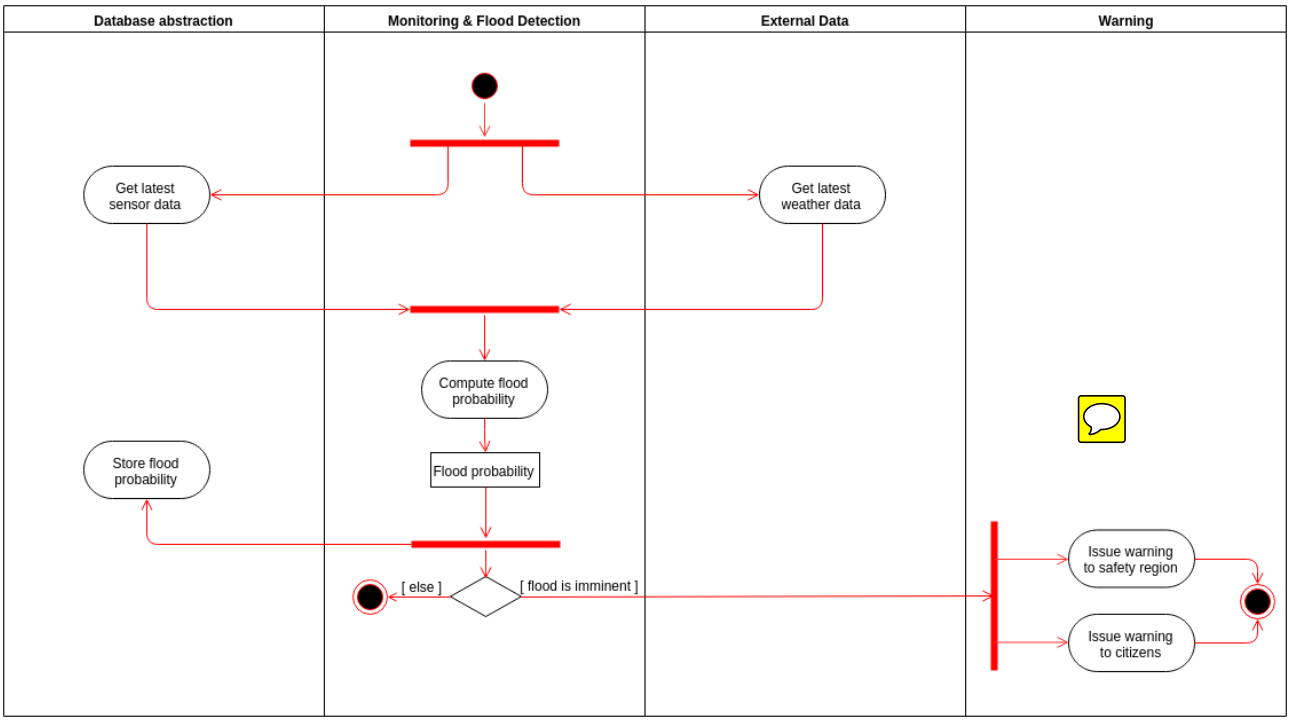


Figure 7.12: An activity diagram of the flood monitoring process

Sending the sensor data

7.2.5 Deployment View

This subsection outlines the physical arrangement of the nodes in a distributed system, the artifacts that are stored on each node, and the components and other elements that the artifacts implement [9]. Communication paths and deploy relationships model the connections in the system.

7.2.5.1 Deployment Diagram

Figure 7.13 shows the deployment diagram of the SFM. The diagram contains several components such as, sensors, UAV, analytics clusters, and database clusters. Main communication between components are done using TCP/IP. All hardware decision are based on chapter 6.

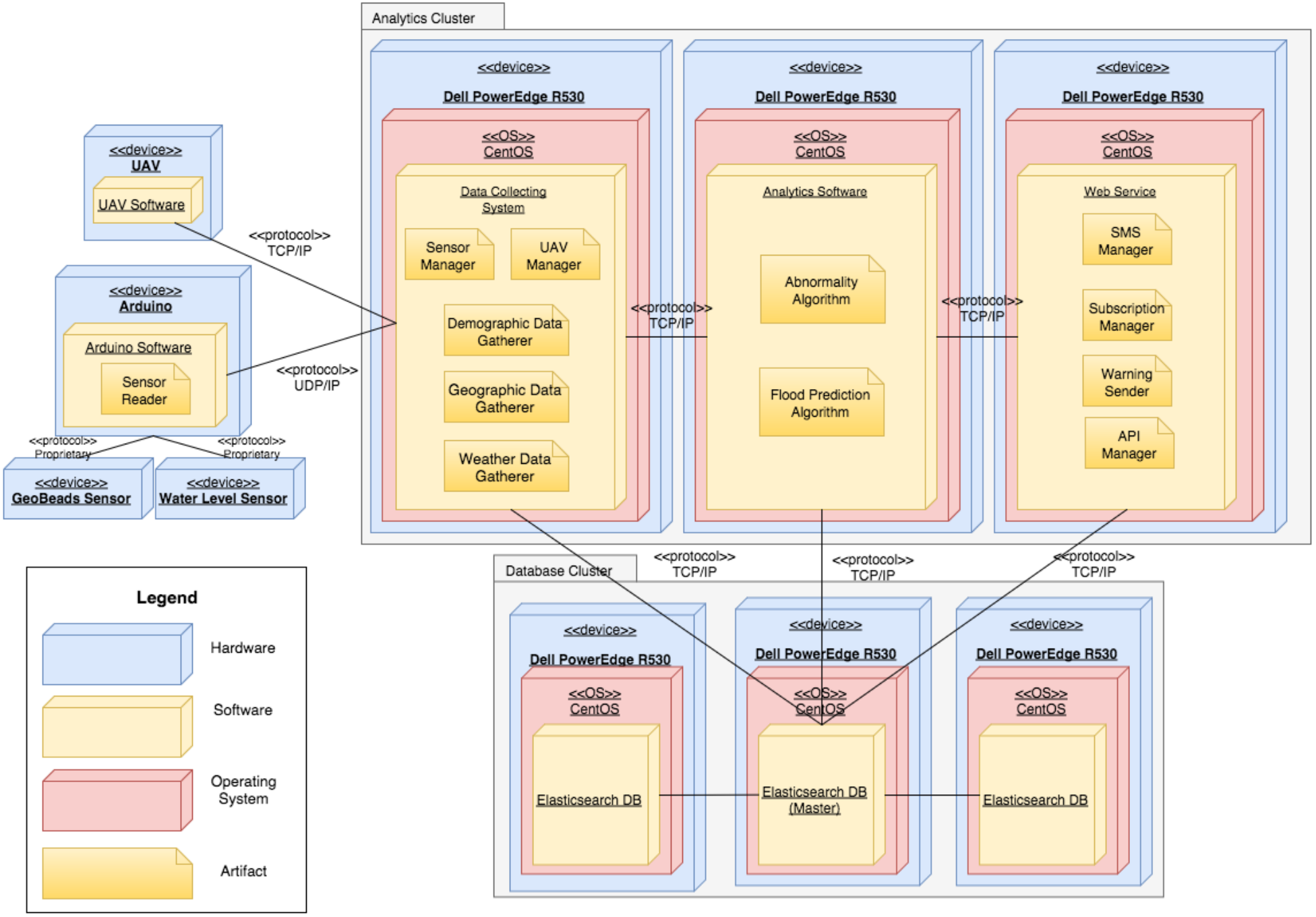


Figure 7.13: Deployment diagram

The deployment diagram, depicted in figure 7.13, is developed based on [9]. The block with blue color represents hardware components. The red and yellow represent operating system and the SFM developed software. Some artifacts are put inside the software block. Deployment view can be categorized into three main components, such as sensing components, main analytics cluster, and database cluster.

Sensing Components

This portion of the SFM is responsible for gathering required data and sending the data to Data Collection System. The SFM will utilize two kind of sensor, water level sensor and GeoBeads sensor. Water level sensor is required to measure the water condition of dykes or water ways, while GeoBeads sensor is used to measure the dykes condition. Both kind of the sensor does not have any software part in it as it is merely electronic component that senses the condition of the surroundings. Both of the sensor will be directly connected to Arduino Uno Ethernet. Arduino will pack the data gathered from sensors and periodically send it to the Data Collecting System.

Furthermore, UAV will fly if it is necessary. UAV will take certain pictures, which is impossible for human to take it, that will be useful for maintenance of the SFM or fixing certain problem.

Main Analytics Clusters

Main analytics cluster is responsible for gathering data and storing it to the database, detecting possible faulty sensors, and analyzing the gathered data to predict imminent flood. The main analytics cluster consist of three main components: data collecting system, analytics cluster, and web service that is open to public.

The SFM will run three data centers to maintain the key drivers of the system. Two of them will be located inside the Netherlands and one will be located abroad. However, figure 7.13 only shows one of them for the sake of simplicity.

Database Cluster

This component will store the data coming from sensors and UAV. The SFM will not store any weather data because the data is always available in the weather provider, thus there is no need to store the data. The SFM will run Elasticsearch database because it is suitable for storing data that the SFM will have.

7.2.5.2 Artifacts

Figure 7.13 contains several artifacts that are derived from Figure 5.2, Figure 7.3, **??**, Figure 7.10, and Figure 7.11. Elaboration of the artifacts will also be categorized into two main components: sensing components and main analytics cluster.

Sensing Components

Artifact	Description
Sensor Reader	Sensor reader is responsible for packing the data gathered from sensor to be ready to be sent.

Table 7.1: Artifact for sensing component

Main Analytics Clusters

Artifact	Description
Sensor Manager	Sensor manager is responsible for managing input coming from multiple sensors (Arduino).
UAV Manager	UAV manager is responsible for handling incoming data from UAVs.

Demographic Data Gatherer	This is responsible for getting data from demographic data provider.
Geographic Data Gatherer	This is responsible for getting data from geographic data provider.
Weather Data Gatherer	This is responsible for getting data from weather data provider.
Abnormality Algorithm	Abnormality algorithm is responsible for detecting faulty sensors.
Flood Prediction Algorithm	This is the main part of algorithm, which will detect imminent flood.
SMS manager	SMS manager is responsible for handling outgoing SMS to certain users.
Subscription Manager	Subscription manager is responsible for managing subscription of citizens/users.
Warning Sender	Warning sender is responsible for sending warnings to Safety region.
API Manager	This is responsible for managing outgoing API to third party developers.

Table 7.2: Artifact for main analytics cluster

7.2.6 Database

The figure 7.14 below shows how the database is structured. 

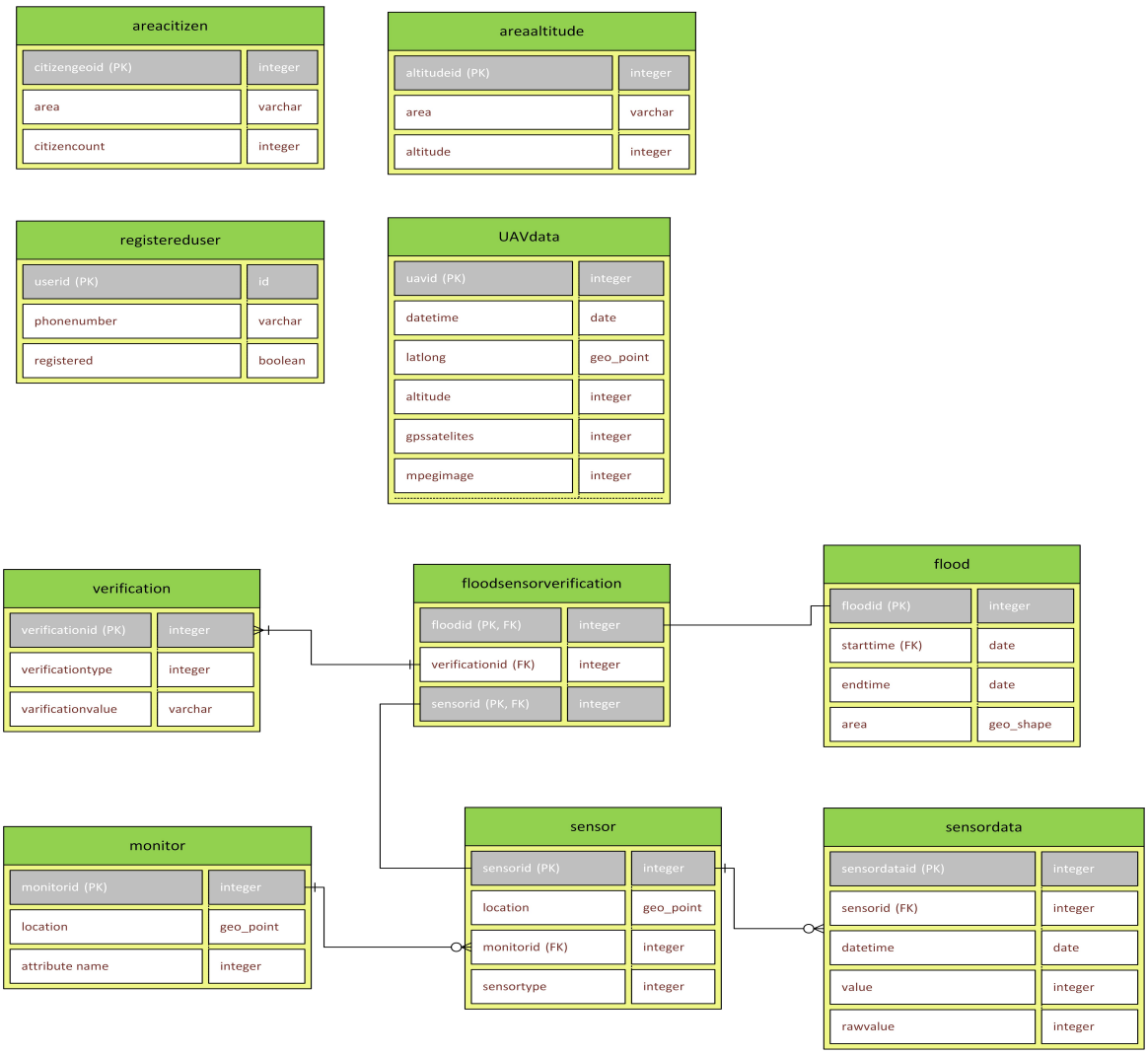


Figure 7.14: Database diagram

The database diagram shows all the information the API can provide to the users.
The database is made of ten tables.

In green is the name of the table.
In grey the primarykey.
In the second column of each table are the types of the data.

The first table **aeracitizen** concerns the area where live the citizens : citizengeold (?) , area : the name of the area and citizencount gives the number of citizen in this area.
The table **aeraaltitude** gives information about the are such as its altitude. The table **UAVdata** provides information sent by the UAVs which will give more information about the flood such as : uavid which identifies a specific uav, datetime, altitude , latlong ,gpssatelite,mpegimage,
The table **registereduser** concerns information to identify a specific user to provide him the warning, provide guidance and help the safety region : his id, phonenumer and if he is registered or not.

Six table are related and provide information about the sensors :
verification (?)
The table **Floodsensorverification** (?)
The table **flood**gives information about the flood : its id by floodid , starttime , endtimide , area.
The table **monitor** : monitorid , its location, attributename.
The table **sensor** : sensorid identifies each sensor, its location , which monitor it depends on,and its type
The table **sensordata** : sensordataid which identifies each data of the sensor, sensorid, datetime which gives the exact time of when the data has been sent ,value, rawvalue(?).

7.3 Software design decisions

This section highlights some of the software design decisions that have been made, especially those that are not covered in the previous views.

Name	Data source architectural pattern
Decision	DEC-5
Problem/Issue	The system components need to be abstract and individually replaceable without having to alter the entire system
Decision	The data mapper [20] is used to completely separate and ignore. the database in the domain model. This provides a more flexible architecture that allows modifications of either the database or the domain model independently without having to alter the other layer.
Alternatives	Table data gateway, row gateway, active record
Arguments	**Table data gateway** The table data gateway is a pattern that, for each domain class, provides a gateway class. This gateway class provides methods to that do all the database interaction. This way all the SQL statements will be stored in the gateway class and thus is separated from the domain layer. **Row gateway** Same idea as the table data gateway, but now for individual rows, instead of a table. **Active record** This is the most basic approach. In this pattern, the domain object also contains the logic of storing that object to the database.

Table 7.4: Decision – Data source architectural pattern

Name	Database
Decision	DEC-6
Problem/Issue	The system needs a database that • Is highly scalable • Has a very good performance • Has a high availability • Has allot of features for storing and querying geospacial data
Decision	The Elasticsearch database is the best database for the SMF system. Elasticsearch is open source, so it will not increase the SMF system costs. It has native geospacial data / querying support [6] and very good geospacial query capabilities [7]. Elasticsearch is extremely distributed and scalable[4]. And Elasticsearch has a very good performance
Alternatives	PostGreSQL, SQL Server, Oracle, MySQL, SOIR, Sphynx, MongoDB
Arguments	**SQL Server** Has the ability to cluster, but only failover clusters [15], so no workload balancing. **Oracle** Has failover clustering options and load balancing options [12], however but costs money. **MySQL** In previous versions only had support for bounding box capabilities, it had no polygon support [11]. The latest version (Version 5.6.1) does have support for polygons [11]. However, MySQL has not yet proven itself to be very good with spacial data. MySQL can be clustered and load balanced by using other tools [8].This, however, is a solution that does not provide the scalability the SMF system is looking for. **SOIR and Sphynx** Are both good choices. Both are able to cluster and distribute and both support geospacial data. However, Elasticsearch scales and distributes better then both [5]. **MongoDB** Has every feature the SFM system needs. It provides allot of support for geodata storing and querying. It is also very good in clustering and distributed querying. However, the performance of Elasticsearch is allot better [10]. **PostGreSQL** is Opensource and has very good geodata support. It supports master-slave clustering, which is not good enough for SFM. However, by using third party applications like pg shard, it is possible to create a cluster of "shards" which is the same technique Elasticsearch uses. This makes PostGreSQL also a good option for SFM. Elasticsearch, however, has a better performance and flexibility.

Table 7.6: Decision – Database

Name	**CentOS - Linux Distribution**
Decision	DEC-7
Status	**Approved**
Problem/Issue	The reliable Linux distribution is needed to run the Smart Monitoring
Decision	The warning system will use CentOS Linux as a platform. CentOS is widely used as server and has many supports.
Alternatives	*Ubuntu* A Debian based Linux distribution that has high installation counts. This distribution is famous for desktop computers. *Fedora* Up-to-date packages, commercial support by third party available, easy to maintain, but lacks on package availability, and hard to do a live upgrade to a newer version.
Arguments	

	Reliability	Resilience	Performance	Interopertability	Security	Scalability	Cost	**Score**
Weight	1	1	1	1	1	1	1	
CentOS	5	4	5	4	5	4	4	31
Ubuntu	4	3	4	4	4	5	4	28
Fedora	3	3	4	4	4	4	4	26

Table 7.7: Decision – Linux Distribution

Name	**Push / Pull for sensor data**
Decision	DEC-8
Status	**Approved**
Problem/Issue	The sensor data has to be sent to the central server
Decision	The sensor will push the sensor data to the central server.
Alternatives	*Push* The sensors send their data using push-technology. *Pull* The server pulls the data from the sensor.
Arguments	**Pull** If the server pulls the data from the sensor an extra request has to be sent to all the sensors, which adds extra overhead. Additionally, the sensor will continuously have to be listening for the servers request. **Push** If the sensors push the data to the server, there is no need for the server to send the extra request, nor do the sensors have to be listening for server request.

Table 7.8: Decision – Push / Pull for sensor data

Name	**Securing sensor data communication**
Decision	DEC-9
Status	**New**
Problem/Issue	The communication pathway between the sensors and the central server has to be secured to prevent unauthorized modification of the sensor data.
Decision	The sensor will use a unique token per sensor, to compute a hash which is send with the sensor data, allowing the central server to confirm the authenticity of the sent data.
Alternatives	*Using unique token* A hash-function is used, taking as input the sensor data and a unique token, producing an output hash, which can be used to verify the data is from the sensor. *Using per-sensor token* Instead of a single token, all the sensors will have their own unique token. This allows revoking the token of an individual sensor. *Using VPN* A VPN (Virtual Private Network) allows all the sensors and the central server to be in the same virtual network. Access to the VPN requires authentication.
Arguments	**Using unique token** The downside of using a unique token is that, when compromised, it requires updating the token on all the sensors. **Using per-sensor token** This requires the central server to have a list of all the tokens. It however, allows the server to revoke one of the tokens if a sensor gets compromised. **Using VPN** A VPN allows communicating over the secure private network. However, the Arduinos which are responsible for sending the sensor data will most likely have problems using a VPN because of their limited CPU and memory.

Table 7.9: Decision – Securing sensor data communication

8 Architecture evaluation

In this chapter the software architecture will be evaluated. This document started with a requirement analysis. The requirements which are defined in chapter 3 are checked if they are fulfilled in the software architecture. Another part in this chapter is the architecture evaluation using Architecture Tradeoff Analysis Method (ATAM). The architectural risks can be exposed that can harm the organization's business goals.

8.1 Requirements validation

The requirements from chapter 3 are evaluated in this section. The section is split according to chapter 3. First the functional requirements, following the commercial requirements, and at the end the technical non-functional requirements.

8.1.1 Functional requirements

Nr.	Priority	Fulfilled	Location	Remarks
FR-1	Must	Yes	7.2.2 6.2	Fulfilled by the system providing a REST server the arduino sensor systems can use to post the sensor data to.
FR-2	Must	Yes	7.2.2	Fulfilled by having the system use several algorithms to analyze the sensor input for abnormalities
FR-3	Must	Yes	6.2 HW-2	Fulfilled by having arduino systems sent the dike sensor values over a cabled network to the central system's REST server
FR-4	Must	Yes		
FR-5	Must	Yes	7.2.3 **??**	Fulfilled by using an ElasticSearch database that is capable of storing all the data needed.
FR-6	Must	Yes	**??**	Fulfilled by using an Elasticsearch database that is very robust, high available and scalable. Thereby enhancing the ability to store and receive data at all times.
FR-7	Must	Yes	5.1.3	Fulfilled by the system contacting several API's to base the decisions on.
FR-8	Must	Yes	7.2.2	Fulfilled by the system using Longitudinal Data Analysis to correlate the sensor data with the forecast data
FR-9	Must	Yes	5.1.3	Fulfilled by the system contacting an external GEO API over TCP/IP
FR-10	Must	Yes	7.3	Fulfilled by the system storing the geographical data and using Elasticsearch queries to get the specified area
FR-11	Must	Yes	5.3 7.2.2	Fulfilled by using several different external API's for additional information together with a correlation algorithm and other analysis algorithms
FR-12	Should			
FR-13	Should			
FR-14	Must			
FR-15	Must			
FR-16	Must			
FR-17	Must			
FR-18	Must			
FR-19	Must			
FR-20	Must			
FR-21	Must			
FR-22	Must			
FR-23	Must			
FR-24	Must			
FR-25	Must			
FR-26	Must			
FR-27	Must			
FR-28	Must			
FR-29	Could			

8.1.2 Commercial non-functional requirements

Nr.	Priority	Fulfilled	Location	Remarks
CNFR-1	Must			
CNFR-2	Must			

8.1.3 Technical non-functional requirements

8.1.3.1 Reliability

Nr.	Priority	Fulfilled	Location	Remarks
REL-1	Must			
REL-2	Must			
REL-3	Must			
REL-4	Must			

8.1.3.2 Availability

Nr.	Priority	Fulfilled	Location	Remarks
AVA-1	Must			
AVA-2	Must			

8.1.3.3 Resilience

Nr.	Priority	Fulfilled	Location	Remarks
RES-1	Must			
RES-2	Must			
RES-3	Must			
RES-4	Must			
RES-5	Must			

8.1.3.4 Performance

Nr.	Priority	Fulfilled	Location	Remarks
PERF-1	Must	Unknown	-	Expected, not yet tested.
PERF-2	Must	Unknown	-	Expected, not yet tested.
PERF-3	Must	Unknown	-	Expected, not yet tested.
PERF-4	Must	Unknown	-	Expected, not yet tested.
PERF-5	Must	Unknown	-	Expected, not yet tested.

8.1.3.5 Interoperability

Nr.	Priority	Fulfilled	Location	Remarks
INTR-1	Must	Yes	5.1.3	Fulfilled by using several weather forecast API providers.
INTR-2	Must	Yes	5.1.2, 5.1.3, 5.1.4	Fulfilled by notifying Safety Region using API.
INTR-3	Must	Yes	5.1.3	Fulfilled by sending SMS to subscribed citizens.
INTR-4	Must	Yes	6.1	Fulfilled by using GeoBeads and Water level sensor.
INTR-5	Must	Yes	5.1.3	Fulfilled by gathering ground height and waterways data.
INTR-6	Must	Yes	5.1.2	Fulfilled by exposing API to third party developers.

8.1.3.6 Security

Nr.	Priority	Fulfilled	Location	Remarks
SEC-1	Must	Yes	5.1.1	Fulfilled.
SEC-2	Must	Yes	5.1.4	Fulfilled.
SEC-3	Must	Yes	6.3.3	Fulfilled, applied in Elasticsearch DB.
SEC-4	Must	Yes	6.1	Fulfilled, in the data center level.
SEC-5	Must	Yes	6.1	Fulfilled, by using Arduino to send data.

8.1.3.7 Scalability

Nr.	Priority	Fulfilled	Location	Remarks
SCALE-1	Must	Yes	6.3.3	Fulfilled by adapting Elasticsearch.
SCALE-2	Must	Yes	6.1	Fulfilled by adapting multiple data centers and multiple sensor type.
SCALE-3	Must	Unknown	-	Expected, not yet tested.

8.2 Architecture evaluation

The ATAM method is used to evaluate the software architecture. The purpose of ATAM is elicit and refine a precise statement of the architecture's driving quality attribute requirements • elicit and refine a precise statement of the architectural design decisions • evaluate the architectural design decisions to determine if they satisfactorily address the quality requirements [add source]

The method consists of the following steps:[add source] 1. Present the ATAM: The evaluation team presents a quick overview of the ATAM steps, techniques used, and outputs from the process. 2. Present the business drivers: The system manager briefly presents the business drivers and context for the architecture. 3. Present the architecture: The architect presents an overview of the architecture. 4. Identify architectural approaches: Itemize the architectural decisions discovered in the previous step. 5. Generate the quality attribute utility tree: Identify, prioritize, and refine the most important quality attribute goals in a utility tree format. 6. Analyze architectural approaches: Probe the architectural approaches in light of the quality attributes in order to identify risks, sensitivity points, and tradeoffs. 7. Brainstorm and prioritize scenarios: Create and analyze scenarios that represent the various stakeholders' interests to understand quality attribute requirements and their relative importance. 8. Analyze architectural approaches: Continue to identify risks, sensitivity points, and tradeoffs while noting the impact of each scenario on the architectural approaches. 9. Present results: Recapitulate the ATAM steps, outputs, and recommendations.

The first three steps are presented in the previous chapters. In this sub-chapter the focus will be on step 4 and further.

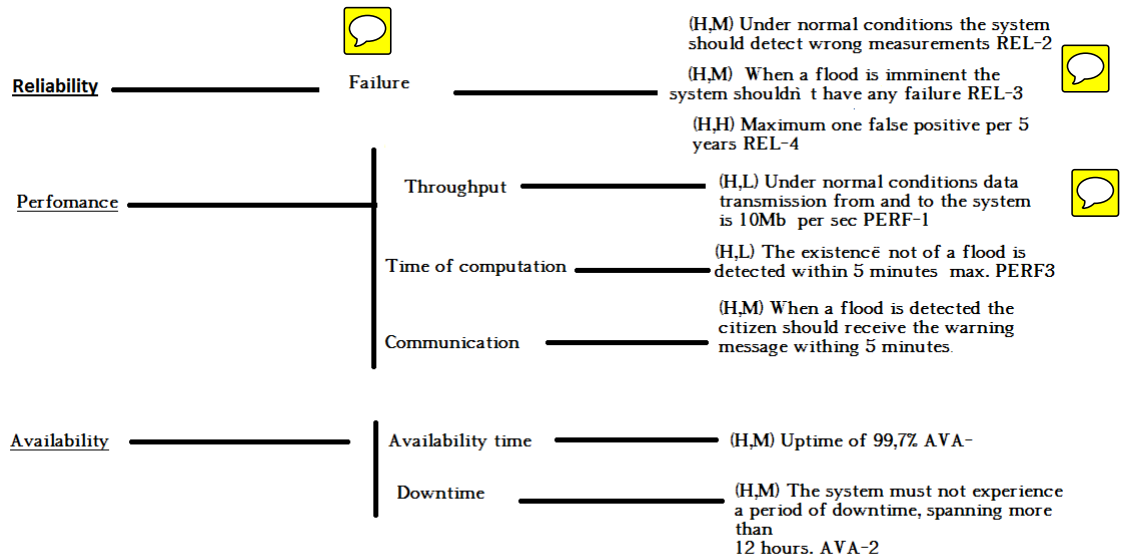
8.2.1 Architectural approaches

8.2.2 Quality attribute utility tree approaches

In this section the utility tree is built according to the key drivers of our system : reliability, performance and availability.

The prioritization is as follow :

Importance to system overall success : High, Medium, Low
Risk/difficulty in achieving : High, Medium, Low



8.2.3 Scenarios 

Scenario	**Handling faulty sensors** 
Q-Attribute(s)	Availability, reliability
Environment	Normal operation
Stimulus	A sensor sends wrong data or stops sending data
Response	The system ignores the sensor until is has been repaired/replaced

Design decisions

Decision	Req.	Sensitiv.	Tradeoff	Risk	Non-Risk
Detection algorithm	FR-18, REL-2	S-1	TO-1		
Reporting	FR-20			R-1	
UAV	FR-30	S-2	TO-2	R-2	

Sensitivities:

- S-1: Some of the time, it will be difficult for the detection algorithm to distinguish between extreme data from a sensor caused by a fault and extreme data caused by a flood.
- S-2: It takes time for the UAV to be dispatched to the potential flood location.

Tradeoffs:

- TO-1: Reliability (+) vs. Performance (-) – Using an algorithm on the sensor data to detect faulty sensors adds more overhead, but increases the reliability of the system.
- TO-2: Reliability (+) vs. Performance (-) – The UAV checks if a flood is present/developing when the sensor data is not conclusive. It has a large dispatch time, which means (if there is a flood), it will be detected with a larger delay.

Risks:

- R-1: There is a risk that the config panel where the faulty sensors are reported, is not checked often enough, leading to broken sensors not being replaced.

Reasoning	Temporarily ignoring faulty sensors allows the system to continue functioning. Reporting the sensors using the control panel allows maintenance personnel to repair those sensors. It is important that maintenance personnel checks the control panel regularly. In case it cannot be determined by the system if a sensor is faulty, or whether there is a flood, a UAV can be dispatched to check.

Table 8.1: ATAM – Handling faulty sensors

Scenario	**Handling hardware failures**
Q-Attribute(s)	Availability
Environment	Normal operation
Stimulus	A hardware component stops operating
Response	The system uses a backup of the hardware component

Design decisions

Decision	Req.	Sensitiv.	Tradeoff	Risk	Non-Risk
Database cluster	AVA-1, AVA-2		TO-3		
Analytic cluster	AVA-1, AVA-2		TO-3		
Multiple data centers	AVA-1, AVA-2		TO-4		NR-1
Arduino	AVA-1, AVA-2	S-3			

Sensitivities:

- S-3: The Arduino are hardware components which can fail as well. Several sensors are connected to a single Arduino. Since a failing Arduino does not have a backup, those sensors connected to it will become unavailable to the system until the Arduino is repaired.

Tradeoffs:

- TO-3: Availability (+) vs. Affordability (-) – Using a cluster is more expensive, but provides a fallback in case of failures, increasing the availability.
- TO-4: Also Availability (+) vs Affordability (-) – The costs of the system increase significantly by having a second data center. However, this guarantees the availability of the system in case of problems with one of the data centers.

Nonrisks:

- NR-1: Multiple data centers aid with increasing the availability only when they are not placed in close proximity to each other.

Reasoning	The hardware in the data centers and the data center itself are prepared for failures of components. It is important to note that the data centers should not be placed in close proximity to each other. A failure of an Arduino will lead to several sensors going offline, but these sensor are located in a relatively small area and therefore only have a limited impact on the systems monitoring capabilities.
Arch. model	See figure 6.2 and figure 6.3 for the logical schematic of the database and analytic cluster respectively. Also see figure 6.1 for an overview of the multiple data centers.

Table 8.2: ATAM – Handling hardware failures

Scenario	**Ensuring availability of third party data / services**
Q-Attribute(s)	Availability, reliability, interoperability
Environment	Normal operation
Stimulus	A third party service or data API becomes unavailable
Response	The system still has access to (a backup of) the data or service

Design decisions

Decision	Req.	Sensitiv.	Tradeoff	Risk	Non-Risk
Using multiple API providers offering same service/data	FR-7, FR-9		TO-5		
SMS service	FR-17, FR-16	S-4			NR-2
Emergency room API	FR-14				NR-3

Sensitivities:

- S-4: The system relies on a single SMS Service provider (CM Telecom). If this provider is not available, the system cannot warn citizens by text message.

Tradeoffs:

- TO-5: Reliability (+) vs. Affordability (-) – Most APIs charge a usage fee, which means that using multiple APIs increases the costs.

Nonrisks:

- NR-2: The phone numbers of citizens who subscribed are stored in the database of the system instead of the SMS Service provider's database.
- NR-3: It is assumed that the emergency room API is always available, which is the responsibility of the safety region.

Reasoning	By using multiple APIs for weather / geographic / demographic data, it is ensured that this data is available if one of the APIs goes offline. The SMS service is a single point of failure: if it goes offline, there is no backup SMS service provider.

Table 8.3: ATAM – Ensuring availability of third party data / services

9 System evolution

A Time Tracking

A.1 Week 1

Person	Task	Hours
Eedema	Reviewing the document, reading the assignment, initializing requirements, & installing environment for project	8
Putra	Initial preparation for the course	5
Fakambi	Reading the document and assignment, Preparation and drafts with ideas	5
Schaefers	Setting up the working environment, create the context page and analysis page drafts. Setting up and improving the the document structure.	8
Klinkenberg		
Brandsma	Creating working environment, reading assignment, first draft business part	8
Menninga	Reading assignment, setting up working environment, first non-functional requirements	5

A.2 Week 2

Person	Task	Hours
Eedema	Coaching session, project planning session and work on business information chapters	9
Putra	Coaching session, project planning session, project meeting, first version of stakeholder part of requirements	7.5
Fakambi	Coaching session , project meeting, work on Non functional requirements	7
Schaefers	First coaching session, improved and enhanced the context and business information chapters. Also created a quality attributes prioritization table.	8
Klinkenberg	Coaching session, meetings, providing feedback on requirements	5.5
Brandsma	First version of use-cases, coaching session, meeting, use-cases, architectural vision	6.5
Menninga	First version of the functional requirements, coaching session, meeting	10.25

A.3 Week 3

Person	Task	Hours
Eedema	Coaching, meetings, analysis, business part, reviewing	14
Putra	Coaching session, meetings, proofread on chapter 1 and 2, revising stakeholders, database decision part of analysis, and preparing LaTeX file for the presentation	10
Fakambi	Coaching session, meetings, Non functional requirements and Risk assessment	10.5
Schaefers	Coaching session, meetings, reviewing, Business section	12
Klinkenberg	Coaching session, meetings, technical requirements, analysis, reviewing	13.5
Brandsma	Coaching session, meeting, architectural vision, use-cases, analysis	9
Menninga	Coaching session, meetings, updates functional requirements, reviewing entire document, updated assumptions and some improvements to structure of analysis, added decision about type of water level sensor.	14.0

A.4 Week 4

Person	Task	Hours
Eedema	Coaching session, meetings, business chapter, review 3, presentations, peer review	13
Putra	Coaching session, meetings, presentation prep., improving business rationale, review chapter 2, making draft of chapter 5, 6, and 7	14
Fakambi	Coaching session , meetings , Work on and improvements chapter 3.5 to 3.8	12
Schaefers	Researched on sensors and other EWS's. Then created the system architecture model diagram and vision diagram for the presentation I had to present in. Created/improved other diagrams. Researched about the costs of these kinds of systems and created/enhanced the business cost section.	20
Brandsma	Coaching session, meetings , reviewing group 1, architectural vision, use-cases, reviewing, improving chapter 4	16.5
Menninga	Coaching session, presentation prep., meeting, improvements FR and risks, review ch. 4, improvements to NFR and Risk Assessment	14.0

A.5 Week 5

Person	Task	Hours
Eedema	Coaching meetings, reviewing, chpt 5	13
Putra	Coaching session, meetings, reviewing, working on initial work on chapter hardware, researching on servers, and working on server selection	15
Fakambi	Coaching session, meetings, Work on chapter 6 Hardware architecture, drafts, design decision about UAVs	12.0
Schaefers	Created initial layer diagram, component diagram, sequense diagram and database design diagram. Researched and explained what new database to use.	15
Brandsma	Coaching session, meetings, chapter 5	13
Menninga	Coaching session, meeting Tuesday, lots of improvements to chapter 3, expanding chapter 6	15.5

A.6 Week 6

Person	Task	Hours
Eedema	Coaching, meetings, review, system overview, evaluation	15
Putra	Coaching session, meetings, reviews, hardware overview, deployment view	13
Fakambi	Coaching session, meetings, Work on chapter 6 Hardware architecture, drafts, design decision about UAVs, draft about chapter 7	9
Schaefers	Meetings. Worked on the implementation and logical section of the software chapter.	13
Brandsma	Meetings, Chapter 5, reviewing	12.5
Menninga	Coaching session, meetings, reviews, hardware overview (sensors)/costs, implementation view, process view	13.0

A.7 Week 7

Person	Task	Hours
Eedema	Creating implementation and the second presentation; Reviewing: process	10
Putra	Coaching session, meetings, updated deployment diagram, added deployment artifacts, added some technical non-functional requirements verification, reviewing	11
Fakambi	Coaching session, work on chapter 7, 8 Utility tree	9
Schaefers	Meetings. Worked on the logical view, updated component diagram	12
Brandsma	Meetings, improved chapter 6, scenarios in chapter 7, looked into architectural evauation	11
Menninga	Meetings, improving software design decisions, adding ATAM tables	11.0

A.8 Todo

Bibliography

[1] Attribute-driven design method. http://www.sei.cmu.edu/architecture/tools/define/add.cfm.

[2] Did you know? http://www.holland.com/global/tourism/article/did-you-know.htm.

[3] Dike monitoring and conditioning system (dmc). http://www.dmc-system.com/.

[4] Distributed nature of elasticsearch. https://www.elastic.co/guide/en/elasticsearch/guide/current/_distributed_nature.html.

[5] Elasticsearch creator reply to performance question. http://stackoverflow.com/questions/2271600/elasticsearch-sphinx-lucene-solr-xapian-which-fits-for-which-usage.

[6] Geo spacial data in elasticsearch. https://www.elastic.co/guide/en/elasticsearch/guide/current/geoloc.html.

[7] Geo spacial querying in elasticsearch. https://www.elastic.co/guide/en/elasticsearch/guide/current/querying-geo-shapes.html.

[8] Ha mysql cluster with nginx and galera. https://www.nginx.com/blog/mysql-high-availability-with-nginx-plus-and-galera-cluster/.

[9] Ibm knowledge center: Deployment diagram. http://www-01.ibm.com/support/knowledgecenter/SS5JSH_9.1.2/com.ibm.xtools.modeler.doc/topics/cdepd.html?lang=en.

[10] Mongodb vs elasticsearch. http://blog.quarkslab.com/mongodb-vs-elasticsearch-the-quest-of-the-holy-perf
html.

[11] Mysql 5.6 spacial functions. http://dev.mysql.com/doc/refman/5.6/en/spatial-analysis-functions.html.

[12] Overview of automatic workload management. http://docs.oracle.com/cd/E11882_01/rac.112/e16795/hafeats.htm#RACAD7120.

[13] Service layer. http://martinfowler.com/eaaCatalog/serviceLayer.html.

[14] Waterwijzer. http://www.citg.tudelft.nl/fileadmin/Faculteit/CiTG/Over_de_faculteit/Afdelingen/Afdeling_watermanagement/Secties/gezondheidstechniek/leerstoelen/Drinkwater/Public/People_interest/doc/Waterwijzer_2004-2005.pdf&usg=AFQjCNEtxg4Dkr1hFahuYM6iiORcgWwEcQ&sig2=8csRJqaHWfFswIz7lUGBiw&cad=rja.

[15] What sql server can and cannot do. http://logicalread.solarwinds.com/what-sql-server-clustering-can-and-cannot-do-w02/#.VhApJHqqpBc.

[16] Make or breaker: Can a tsunami warning system save lives during an earthquake? http://www.scientificamerican.com/article/can-tsunami-warning-system-save-lives-eartquake/, 2011.

[17] 2004 indian ocean tsunami 10 years later: Warning system installed after disaster has 'critical gaps'. http://www.ibtimes.com/2004-indian-ocean-tsunami-10-years-later-warning-system-installed-after-disaster-has-1763662, 2014.

[18] Natural disasters cost the world over 300 billion per year on average. http://www.dw.com/en/natural-disasters-cost-the-world-over-300-billion-per-year-on-average/a-18316637, 2015.

[19] Steady increase in climate related natural disasters. http://www.accuweather.com/en/weather-blogs/climatechange/steady-increase-in-climate-rel/19974069, 2015.

[20] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[21] J.R. Moore. *Application of the Self-Potential Method in Hydrogeology*. PhD thesis, University of California, 2007.

[22] Gordon Ng and Kyle Oswalt. Levee monitoring system. *Better Management through Better Information. CEE*, 180:1–12.

[23] T.N. Spaargaren. *Effectiveness of sensors in flood defences.* PhD thesis, Delft University of Technology, 2012.

[24] M. Van der Geest and Kok S. Draadloos sensornetwerk in dijken. Bachelor thesis, TU Delft, 2009.

[25] K. Wiegers and J. Beatty. *Software Requirements.* Developer Best Practices. Pearson Education, 2013.