Software Architecture

Remote Measurement, Monitoring and Control System: Phase I&II - Domain analysis and requirement elecitation and Architectural Design

Professoren: Wouter Joossen,
Riccardo Scandariato,
Dimitri Van Landuyt,
Kim Wuyts

Maarten Allard - s0199048
Kristof Coninx - s0199831

May 12, 2012

Contents

List of Figures

1 Preface

For the creation and documentation of this system, it was recommended to use Visual Paradigm as the design editor of choice. However due to some compatibility issues that arose from trying to use Visual Paradigm as the editor between different members of the design team, Another software suite was used for creating the design diagrams. We did however consult the didactical team about using another diagram design tool than Visual Paradigm. We also made sure that the software program we used adhered to the same uml specifications as Visual Paradigm such that the diagram contents would be readable and understandable by anyone who is used to Visual Paradigm.

The design tool that was eventually used to document the development of the ReMeS system is called UMLet, a free, open-source uml tool. We took the time to confirm that the diagrams actually are similar to diagrams generated from Visual Paradigm. The only noticeable difference is that in Visual Paradigm, when drawing components, both the component symbol and stereotype (¡¡component¿¿) are shown. However in our diagrams, components are only represented by rectangles with the component-symbol in the top right corner. These are not to be confused with single blank rectangles which are used to indicate class elements. Our specification is still conform to the uml specification about representing components, as is Visual Paradigm's.

2 Overview

This report handles the domain analysis, the functional and non-functional requirements for phase I of the `Remote Measurement, Monitoring and Control System`. We will handle, in order, the domain model for the Remote Metering domain, additional constraints, a glossary, an overview of the use cases in the form of a use case diagram and the detailed use case scenarios. Finally we will also handle some important quality attribute scenarios.

3 Domain Analysis

3.1 Domain components

To make it more easy to process and discuss the domain model for remote metering, the domain can be decomposed into different components. The diagram itself has been provided in full, but the different components can be discerned by their color.

The blue color marks the main component of the remote metering domain. This entails the elements representing the modules and the people using them. These component is quite tightly coupled to the next component depicted in orange.

The orange color marks the utility companies and all elements that are directly related to the utility companies.

The red color marks the anomaly-component. All domain elements that have to do with anomalies, problems, alarms and such, are part of this component.

The green color represents the final component and marks all elements which are related to other external parties not related to the immediate metering domain. These elements include external experts, telecommunications providers and such.

NOTE: According to a teaching assistant, the domain model should also contain type information for the attributes of entities in the domain model. Therefor we provided type information in the domain model. We are, however, of the opinion that the domain model is not the place to include type information and would not have done so if not directed by a TA. Type information for attributes is, in our opinion, more a solution oriented aspect instead of a problem oriented aspect.

3.2 Domain model diagram

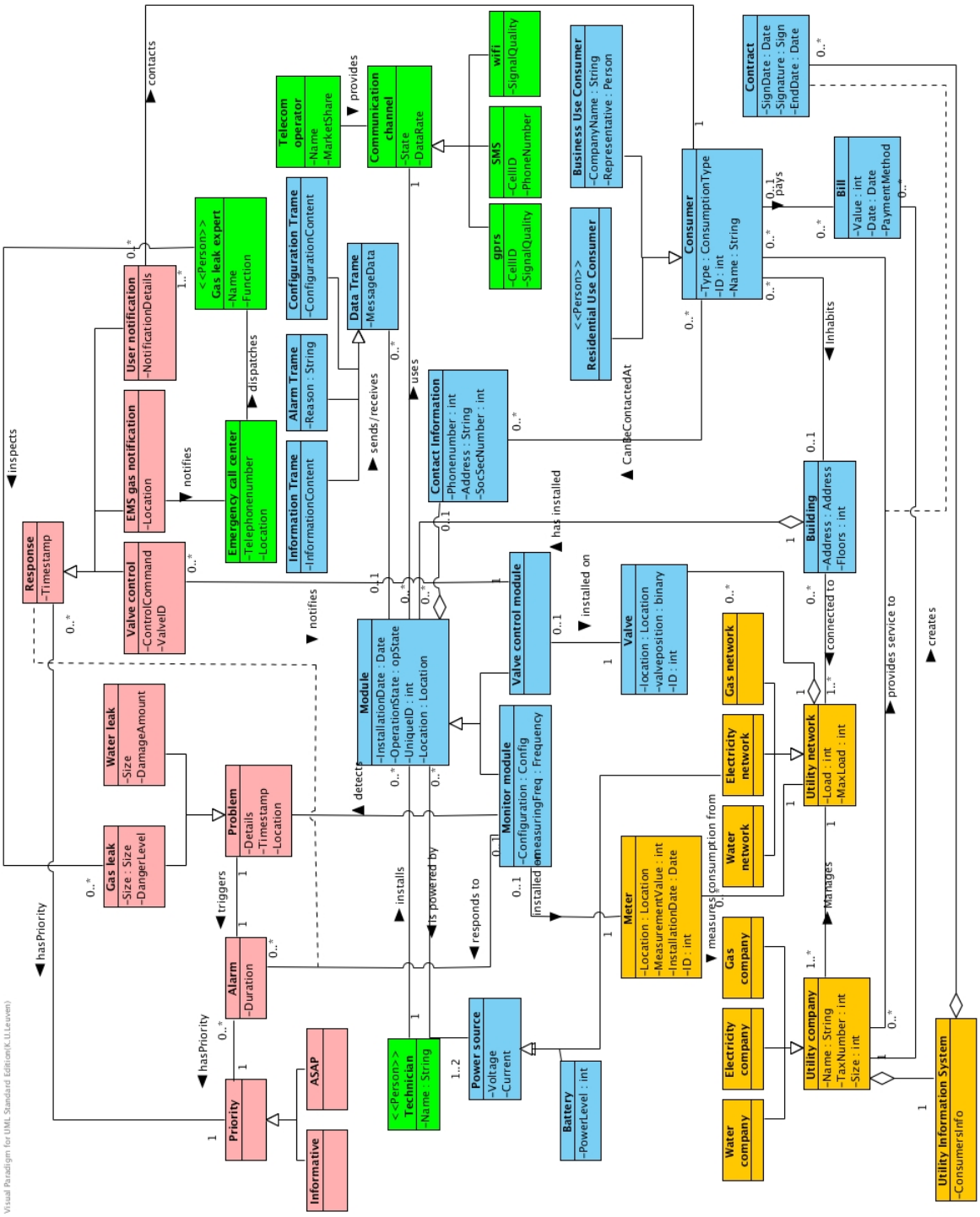

Figure 1: Full domain model with colored components.

3.3 Additional Domain Constraints

In this section some extra domain constraints will be specified in the formal OCL. It was not possible to directly model this information in the domain model, so therefor the following OCL constraints were created.

```
context Consumer:
        self.canBeContactedAt.Address = self.inhabits.Address

context Monitor module inv:
        self.location = self.EMSGasNotification.location

context Monitor module inv:
        self.detects.oclIsKindOf(Gas Leak) implies
        self.Response.asSet()->includes(EMSGasNotification)

context Monitor module inv:
        self.respondsTo.hasPriority.oclIsKindOf(self.Response.hasPriority)

context Monitor module inv:
        not self.detects.asSet()->isEmpty() implies
        not self.respondsTo.asSet()->isEmpty

context Utility company inv:
        self.oclIsKindOf(Water company) implies
        self.manages.oclIsKindOf(Water network)

context Utility company inv:
        self.oclIsKindOf(Elektricity company) implies
        self.manages.oclIsKindOf(elektricity network)

context Utility company inv:
        self.oclIsKindOf(Gas company) implies
        self.manages.oclIsKindOf(Gas network)

context Contract inv:
        self.SignDate < self.EndDate

context Alarm inv:
        self.Duration > 0

context Module inv:
        self.oclIsKindOf(Valve control Module) implies
        (self.isPoweredBy.oclIsKindOf(Battery) and
        self.isPoweredBy.oclIsKindOf(Electricity network))

context Utility network inv:
        self.Load < self.MaxLoad
```

4 Glossary

In this glossary section, some non trivial domain elements will be furtherly explained. The elements will be explained per domain component.

4.1 Main Metering (Blue)

4.1.1 Module

The Module represents the remote measurement modules and the remote valve control modules. These modules are installed on Meters and valves respectively. These modules are also powered by a Power source. This can either be a battery or the electricity network provided by the utility network described in the Utility Company section.

4.1.2 Valve

The valve the valve control module is installed on, is part of the main metering component and not part of the utility company component. This is because the valve control and valve itself are often delivered in one single component by the people in charge of the metering. The context for usage of this element lies not with the Utility Company.

4.1.3 Data Trame

The data trame entity is an entity that is used to communicate with modules. There is an hierarchy of trames in existence that is structured on the type of trame. As such there is an information trame, an alarm trame and a configuration trame.

4.1.4 Contact Information

The contact information entity represents the information that a module is configured to have about who to notify in case of an emergency.

4.1.5 Consumer

The consumer represents the person or company that uses or consumes the services offered by utility companies. This consumer inhabits a building, whether it be a personal home residence or perhaps a company facility.

4.1.6 Contract

When a utility company provides services (in the form of either gas, electricity or water) a contract is signed between the utility company and the consumer. This contract holds all the information about the details, terms, conditions and duration of the agreement between consumer and provider.

4.2 Utility Company (Orange)

4.2.1 Meter

The Meter is an entity representing the physical meter a utility company installs on various locations in the utility network to monitor the consumption of utility resources. These meters are installed with or without remote measurement modules.

4.2.2 Utility Network

The utility network entity represents the different utility networks in the domain. It is a high level representation of the gas, electricity and water networks on a large scale. The networks have collections of valves and meters that are installed in various locations in the network. In the functional requirements analysis section, the term utility network interface point will occur. This term is merely a description of a location where a valve or a meter can be installed in a network.

4.2.3 Utility Information System

The utility Information System is an existing information system that is present in all utility companies. Each utility company has such a system for storing contact information and contracts of customers.

4.3 Anomalies (Red)

4.3.1 Problem

The problem entity represents a problem such as a possible leak. This problem can be detected by the remote monitoring module. When a problem is detected, an alarm will be triggered with an appropriate response. The possible problems the remote monitoring module can detect at this moment are a gas leak and a water leak, depending on what utility network the monitoring module is installed. A problem also has a timestamp, indicating when the problem was detected.

4.3.2 Alarm

Whenever a problem is detected, an alarm will be triggered for this problem. An alarm has an appropriate response which will be the response of the monitor module. A monitor module responds to an alarm. Each alarm also has a priority. A gas leak has a higher priority as a water leak for example.

4.3.3 Response

Each problem has an appropriate response. Responses will also receive the same priority as the alarm of the problem that causes them. Possible responses are valve control, where the monitor module will instruct the valve control to close the valve, user notification, where the monitor module will inform the user from its contactinformation about the alarm, and EMS gas notification, where the monitor module will alert the Emergency Services about a gas leak. This last response can only be used in case that the problem is a gas leak.

4.3.4 Priorities

A response and an alarm have a priority. Connected alarms and responses have the same priority. A priority is an instance to indicate how fast an alarm must be dealt with or how fast the response must be executed. Possible priorities are ASAP and Informative.

4.4 Other External Parties (Green)

4.4.1 Emergency Call Center

An emergency call center is a center someone or something can call in case of emergency. The monitor module uses this call center through a EMS gas notification when a gas leak is detected.

The monitor module will contact the emergency call center, notifying the authorities about the gas leak. The emergency call center will respond to this call by dispatching one or more gas leak experts who will inspect the gas leak.

4.4.2 Communication Channel

A communication channel is an external communication channel that the remote module will use to send or receive data trames. To use this communication channel, there has to be made an agreement with a telecom company about the services it can offer. Without this contract the remote modules won't be able to use the external communication channel. A communication channel can be a wifi connection, a gprs connection or a SMS service.

4.4.3 Technician

A technician can be a regular plumber who knows how to remove and install remote modules. This plumber or techician has to be certified to install or deinstall certain modules.

5 Functional Requirements

5.1 Use Case Diagram

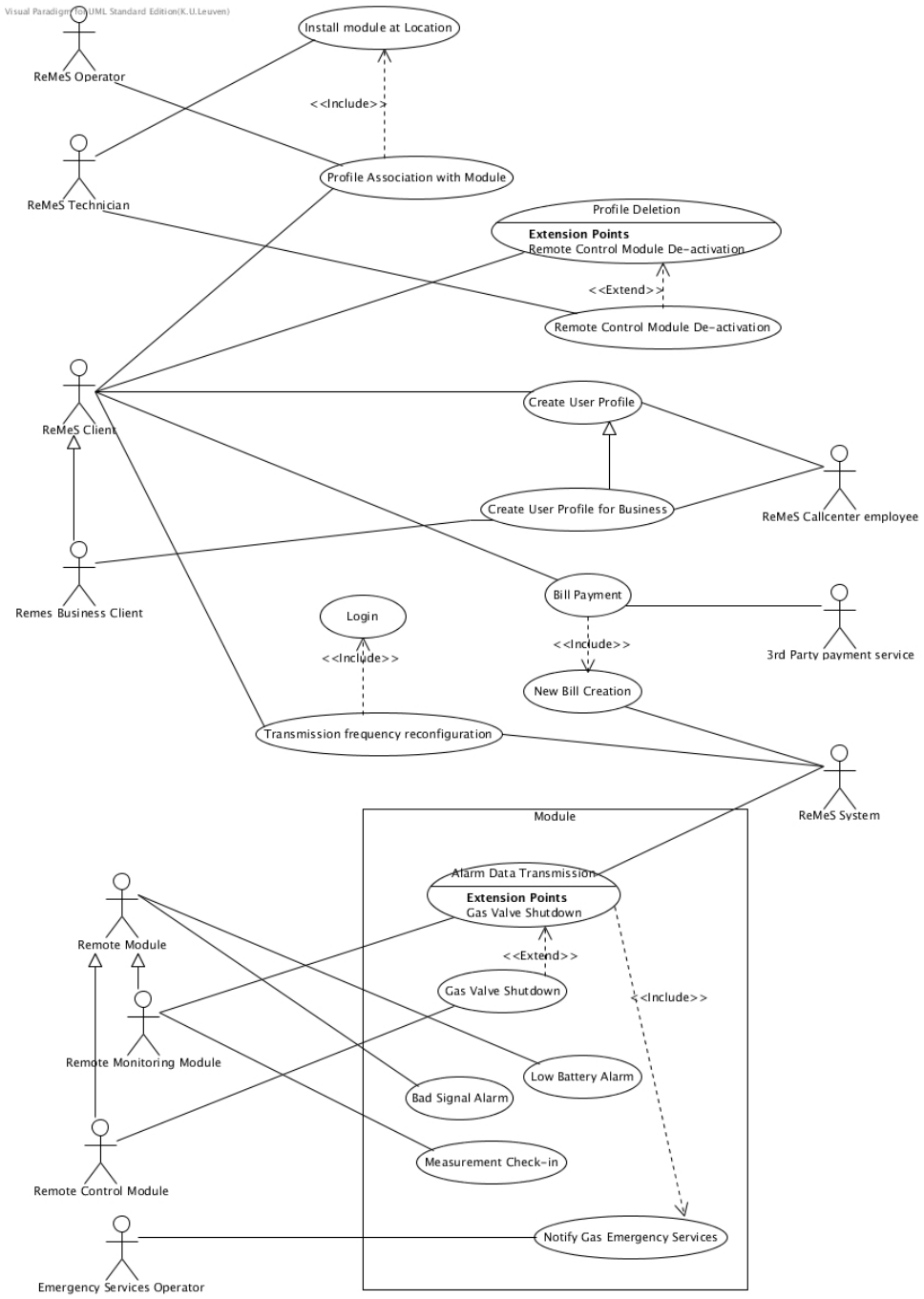

Figure 2: Use Case Diagram for ReMeS System.

5.2 Use Case Scenarios

In this section, the use case scenarios from the use case diagram will be explained. The official standard for inclusion, extension and generalization have been used. Use case generalization will be done by copying the base use case and marking the changes in bold.

5.2.1 Create user profile

Use Case Name

Create user Profile

Primary Actor

A potential client: Whenever someone wants to join ReMeS, the future client first needs to request an account.
ReMeS call center employee: Whenever a potential client calls to request a new profile, the ReMeS operator needs to send an empty contract to the potential client.

Interested Parties

Utility Company: The utility company of which the potential new client of ReMeS is a client of.

Preconditions

The person in question can not already be a client of ReMeS.

Normal Flow

1. A person calls the ReMeS call center and requests the contract for becoming a residential client to be sent to his home address.
2. A contract is sent to the provided home address.
3. The person fills in the contract and sends it back to ReMeS.
4. The person is added to the ReMeS system as a residential user.
5. A confirmation letter is sent to the new user's home address by regular mail.
6. The new client receives a letter containing an authentication token consisting of a user name and password.

Alternative Flow

4a. The person filled in the wrong information and the contract is not valid.

- 4a1. The person is sent a letter explaining that there was a problem processing his request. The person is also sent a new contract to be filled in.
- 4a2. Goto step 3.

Postcondition

The person in question is now a client of ReMeS and can login with his personal user name and password.

5.2.2 Create user profile for business user.

Use Case Name

Create user Profile for **business** user.

Primary actor

Potential client: An employee of a company wanting to become a client of ReMeS.
ReMeS call center employee: Whenever a potential client calls to request a new profile, the ReMeS operator needs to send an empty contract to the potential client.

Interested Parties

Utility Company: The utility company of which a client wishes to become a client of ReMeS.

Preconditions

The company in question can not already be a client of ReMeS.

Normal Flow

1. An **employee of a company** calls the ReMeS call center and requests the contract for becoming a **business** client to be sent to his address.
2. A contract is sent to the provided address.
3. The **employee** fills in the contract and sends it back to ReMeS.
4. The **company** is added to the ReMeS system as a **business** user.
5. A confirmation letter is sent to the **company**'s address by regular mail.
6. The **company** receives a letter containing an authentication token consisting of a user name and password.

Alternative Flow

4a. **The employee** filled in the wrong information and the contract is not valid.

4a1. **The employee** is sent a letter explaining that there was a problem processing his request. **The employee** is also sent a new contract to be filled in.

4a2. Goto step 3.

Postcondition

The **company** is now a client of ReMeS and can login with his personal user name and password.

5.2.3 Profile Association

Use Case Name

Profile Association with remote monitoring module

Primary Actor

ReMeS Operator: An operator at ReMeS who links the device to the user profile.
Client representative: A residential user or a representative for a business user.

Interested Parties

Utility company: The Utility company's UIS wants to be informed of the outcome of this use case.
Regular Mail Service: The regular postal service which will transport and deliver the modules.

Preconditions

The representative who will be provided a remote monitoring module has to be a residential, or a business client of ReMeS.

Normal Flow

1. The use case starts when a ReMeS operator gets the order to link a profile to a module.
2. The ReMeS operator verifies that the monitoring module has an identification number that has not been used yet in the system.
3. The ReMeS operator creates a new entry in the system for the remote monitoring module.
4. The ReMeS operator links the newly created entry to the user profile provided.
5. The ReMeS operator sends the module by regular mail to the address provided by the user profile
6. The client representative receives the module.
7. Include use case *Install module at location.*
8. The ReMeS operator receives confirmation from the system that installation has been successful.
9. The ReMeS operator signals the UIS of the utility company in question that the customer in question now uses ReMeS.

Alternative Flow

5a. In the user profile it is indicated that the user would rather pick up the module at a ReMeS collection point.

   5a1. The ReMeS operator sends the module by regular mail to the ReMeS collection point indicated in the user Profile.

   5a2. The client representative goes and picks up the module from the collection point.

Postcondition

The module is installed at the site and is linked to the user profile the module was intended for.

5.2.4 Install Module

Use Case Name

Install Module at location

Primary actor

ReMeS Technician: A certified technician that install the module.

Interested Parties

Client: The client owning the module which will be installed.
ReMeS operator: The ReMeS operator who needs confirmation on the succesful installation of the module.

Preconditions

The module cannot already be installed. The appointment for the technician is verified and the client has been reminded of the appointment.

Normal Flow

1. The ReMeS technician arrives at the location where the module is supposed to be installed.
2. The technician installs the module physically on the utility net interface.
3. The technician physically seals the module to prevent tampering.
4. The technician enters the current meter value in the system.
5. The module uses the pre-cooked configuration to communicate with the ReMeS system and sends its initial trame to the system.
6. The ReMeS system acknowledges the correct installation of the module and notifies interested parties of this event.
7. The technician finalizes the paperwork and asks for a signature.

Alternative Flow

None

Postcondition

The module is installed and works correctly at the desired location. The interested parties are also notified of the installation of the module.

5.2.5 Measurement Check-in

Use Case Name

Measurement check-in data transmission

Primary actor

The remote measurement module: Measures the data and initiates the check-in.
ReMeS system: receives the data.

Interested Parties

Utility company: Is interested in the data gathered.

Preconditions

The module is powered (either by battery or electricity net).

Normal Flow

1. The use case starts when the module is powered on or when this use case has been completed.
2. The module reads the preset configuration for the desired interval.
3. The module sets a timer matching the interval.
4. The set timer expires
5. The module sends a trame containing the measurement data to the ReMeS system.
6. The ReMeS system receives and stores the measurement data from the module.

Alternative Flow

None

Postcondition

The measurement is sent to and stored at the ReMeS System in specified intervals.

5.2.6 Log in

Use Case Name

Log in to ReMeS.

Primary Actor

ReMeS Client: The client who wants to login.
ReMeS System: The system who handles the procedure.

Interested Parties

ReMeS: Is interested in the people who login to their service.

Normal System Flow

1. The system asks for username and password.
2. The customer provides username and password.
3. The provided username exists.
4. The supplied password for the username is correct.
5. The system loads the preferences of the customer.

Alternate System flow

3a. The username doesn't exist.

3a1. The customer decides to try again. GOTO step 1.

4a. The password is incorrect.

4a1. The customer decides to try again. GOTO step 1.

Outcome (post condition)

The Customer is logged into the store.

5.2.7 Transmission frequency reconfiguration

Use Case Name

Transmission frequency reconfiguration

Primary actor

ReMeS client: The client of the ReMeS system ordering the reconfiguration.
ReMeS system: Handles the order.

Interested Parties

Utility company: Can offer lower prices for consumers who update more frequently.

Preconditions

The client has a remote monitoring module installed.

Normal Flow

1. The use case starts with the ReMeS client deciding to change the update frequency.
2. include use case *Log into ReMeS*.
3. The ReMeS system show the user a list of installed modules.
4. The client selects the module he wants to reconfigure.
5. The client indicates that he is willing to send data more frequently.
6. The ReMeS system processes this indication and sends a configuration command (trame) to the module in question.
7. The previously chosen module receives a reconfiguration command correctly.
8. The module processes the command and sets its update frequency accordingly.
9. The module sends an acknowledgement back to the ReMeS system.

Alternative Flow

4a. The selected module is not a valid module for reconfiguration.

4a1. The user is notified. GOTO step 3.

7a. The module does not receive the command correctly.

7a1. The module does not do anything and does not send back an acknowledgement.

7a2. After a fixed amount of time a system timer at the ReMeS system expires and the event is logged.

7a3. The command is sent again. GOTO step 7.

Postcondition

The update frequency of the module has been succesfully changed at the clients intent. The module will update more frequently from now on.

5.2.8 Alarm data transmission

Use Case Name

Alarm data transmission for the gas monitoring module.

Primary actor

Remote monitoring module (gas): Notices an anomaly and does the notifying.
ReMeS system: The system needs to notify interested parties and emergency services.

Interested Parties

ReMeS Client: will be notified of the anomaly.
Emergency services (gas): Will be notified of gas anomalies.

Preconditions

The module in question needs to be a gas remote control module.

Normal Flow

1. The use case starts when the module detects an anomaly
2. The module ignores all timers and pending tasks and immediately sends an alarm trame to the ReMeS system.
3. The ReMeS system receives the alarm trame from the module.
4. The ReMeS system looks up the contact information that is related to the module.
5. The ReMeS system sends an alarm notification to the recipient, as configured in the remote module.
6. include use case *Notify Gas Emergency Services*
7. Extension point: Use Case *Gas Valve shutdown* if the client has a gas remote control module installed.

Alternative Flow

4a. The module is configured not to send notifications to the client.

4a1. Goto step 6.

Postcondition

The emergency services are notified and the alarm notification of the user is carried out if configured.

5.2.9 Gas Valve Shutdown

Use Case Name

Gas Valve Shutdown

Primary actor

ReMeS system: The system that commands the valve shutdown.
Remote control module: The module that shuts down the valve.

Interested Parties

The ReMeS client: The client might be hindered by the operations described.

Preconditions

There must be a remote control module installed. The control command sent must be a shutdown command.

Normal Flow

1. The use case starts when the remote control module receives a control command indicating the initiation of a valve-shutdown.
2. The module parses the content of the control command.
3. The module effects the mechanical shutdown of the valve.
4. The valve was successfully shutdown. The module send an acknowledgement that shutdown was successfully executed.

Alternative Flow

4a. Something went wrong in shutting down the valve.

4a1. The module sends a negative acknowledgement to the ReMeS system indicating that something went wrong in shutting down the valve and further action should be taken.

Postcondition

An attempt is made to shut down the valve and the ReMeS system is notified of the result of the attempt.

5.2.10 Bill Payment

Use Case Name

Bill Payment

Primary actor

Client: The client that has to pay the bill.
3th party payment service: The 3th party payment service that has to manage the bill.

Interested Parties

ReMeS: ReMeS must create bills for clients at certain fixed points.
Utility company: The utility company wants to know how it will receive its payments.

Preconditions

The utility company has chosen to let ReMeS handle its payments.

Normal Flow

1. Include use case *New Bill Creation*.
2. The client receives the bill created by ReMeS.
3. The client pays the bill to the 3th party payment service
4. The 3th party payment service pays the utility company accordingly.
5. The 3th party payment service notifies ReMeS that this bill is paid.
6. ReMeS updates this information appropriately.

Alternative Flow

3a. The client waits too long to pay the bill

- 3a1. The 3th party payment service sends a reminder notification to the client.
- 3a2. The client receives the notification
- 3a3. Goto step 3.

Postcondition

ReMeS has indicated that this bill is paid.

5.2.11 New Bill Creation

Use Case Name

New Bill Creation

Primary actor

ReMeS: ReMeS will have to create new bills at certain fixed points.

Interested Parties

Client: The client of the utility company will want to know how its bills are created.
Utility company: The utility company will want to keep track of the bills that it still need to receive.
Third party payment service: The third party payment service will want to know what bills it needs to keep track of.

Preconditions

The utility company has chosen to let ReMeS handle its bill creations.

Normal Flow

1. ReMeS has received enough measurement data
2. ReMeS creates a bill for the client
3. ReMeS notifies the utility company and the third party payment service about the creation of the bill.
4. The bill is sent to the client.

Postcondition

A bill is created for the client to pay for the utilities he/she consumed.

5.2.12 Remote Control Module De-activation

Use Case Name

Remote Control Module De-activation

Primary actor

ReMeS Technician: A technician who removes the remote control module
Client: A client who needs his/her remote control module removed.

Interested Parties

None

Preconditions

The client must have a remote control module installed. The appointment for the technician is verified and the client has been reminded of the appointment.

Normal Flow

1. A ReMeS technician arrives at the location where the module is installed.
2. The technician removes the remote control module from the utility net interface.
3. The technician notifies ReMeS that the remote control module has been removed.
4. ReMeS updates his information that the remote control module is no longer active.
5. The technician finalizes the paperwork and asks for a signature.

Alternative Flow

1a. There is nobody at the location where the module is installed who can grant access to the technician.

1a1. The technician notifies ReMeS that a new appointment must be made.

1a2. The technician starts a new assignment.

1a3. ReMeS requests the client to make a new appointment.

1a4. The client makes a new appointment.

1a5. The client is charged a fee for missing the appointment.

1a6. Goto step 1.

Postcondition

The remote control module is removed from the location.

5.2.13 Low Battery Alarm

Use Case Name

Low Battery Alarm

Primary actor

Remote monitoring module: The module that has a battery.

Interested Parties

ReMeS technician: A ReMeS technician sometimes needs to help a client to replace the battery of a remote module.
Client: The client will want to know how and when to replace the battery of a remote module.

Preconditions

The client has a remote monitoring module installed

Normal Flow

1. The remote monitoring module's battery is low
2. The remote monitoring module sends a low battery alarm to ReMeS

3. ReMeS receives the alarm and creates an alarm to the appropriate recipient
4. The recipient receives the alarm and replaces the battery
5. The next time the remote monitoring module sends data, the battery status is automatically updated.

Alternative Flow

4a. The recipient is unable to replace the battery

4a1. The recipient notifies ReMeS.

4a2. A ReMeS technician will be sent on site.

4a3. The ReMeS technician replaces the battery.

4a4. Goto step 5.

Postcondition

The remote module has a new, fully charged battery.

5.2.14 Bad Signal Alarm

Use Case Name

Bad Signal Alarm

Primary actor

Remote monitoring module: The module that works with a remote connection.

Interested Parties

ReMeS technician: Sometimes a ReMeS technician must go on site to solve the problem with a remote module.
ReMeS operator: If a remote monitoring module hasn't been repaired within a certain period of time, a ReMeS operator must call and help the client to repair the problem.
Client: A client will be notified and will try to fix the problem with the bad signal of the remote module.

Preconditions

The client has a remote monitoring module installed

Normal Flow

1. ReMeS detects that a remote monitoring module hasn't sent any data for a certain period of time.
2. ReMeS creates an alarm.
3. ReMeS notifies the client and the ReMeS call center about this alarm.
4. The client solves the problem with the module.

Alternative Flow

4a. The client is unable to solve the problem.

4a1. A ReMeS operator contacts the client.

4a2. The ReMeS operator tries to look for the problem and its resolution.

4b. The client and the ReMeS operator are unable to solve the problem.

4b1. A ReMeS technician is dispatched to the user.

4b2. The ReMeS technician solves the problem with the remote monitoring module.

Postcondition

The bad signal problems with the remote module are now solved.

5.2.15 Profile Deletion

Use Case Name

Profile Deletion

Primary actor

Client: A client who no longer wishes to be a client

Interested Parties

ReMeS: ReMeS will want to know how and when to delete profiles.

Preconditions

The client is a residential or a business user of ReMeS.

Normal Flow

1. A client notifies ReMeS that it wishes no longer to be a client of ReMeS.
2. Extension Point: use case *Remote Control Module Deactivation*.
3. ReMes indicates that the profile of the client is now inactive.
4. The client is notified about his/her profile deletion.

Alternative Flow

None

Postcondition

The client is no longer a residential or a business user of ReMeS.

6 Quality Attribute Scenarios

6.1 Availability

6.1.1 Av1: Communication channel between the remote module and the ReMeS system

Because of a failure in the intermediate telecom infrastructure, or because of a bad signal from the remote module to the intermediate telecom infrastructure, key functionalities of the ReMeS system are compromised: reading data from a remote monitoring module cannot be sent to ReMeS and ReMeS cannot control the remote valves in case of a leak.

- **Source:** external
- **Stimulus:** The external communication channel between the remote module and the ReMeS system is failing. This results in the inability of sending data from ReMeS to a remote valve or from a remote monitoring module to ReMeS. It is also possible that there is a bad signal between the remote module and the external communication channel of the intermediate telecom infrastructure.
- **Artifact:** Remote modules, external communication channel of the intermediate telecom infrastructure.
- **Environment:** normal execution
- **Response:**
  - Prevention:
    - The ReMeS company has negotiated a Service-Level Agreement (SLA) with the intermediate telecom operator that maintains the external communication channel. This SLA states that the external communication channel has 98% availability.
    - Before installing the remote module, the connection to the external communication channel should be checked if it's sufficient. Otherwise another solution needs to be found to improve this connection such as trying another type of connection (gprs, wifi, SMS).
  - Detection:
    - ReMeS detects that a remote module hasn't sent any data for a certain time.
    - ReMeS detects that all remote modules using a certain type of communication (gprs, wifi, 3G) haven't sent any data for a certain time. ReMeS can conclude that the external communication channel is failing and not the signal between the remote module and the external communication channel.
    - ReMeS keeps track of how long there has been a lack of communication.
  - Resolution:
    - In case that one remote module hasn't sent any data for a certain time, the client owning the remote module is notified. Also a ReMeS operator is notified. If the problem isn't solved in time, the ReMeS operator will contact the client. If the problem still isn't resolved after the help of the ReMeS operator, a ReMeS technician will be sent to solve the problem with the remote module.
    - In case that the external communication channel is failing, the ReMeS System Administrator is notified. The System Administrator will contact the telecom operator to resolve this problem.
- **Response measure:**
  - Detection time

    - The detection time of a single module equals to the transmission rate of the remote monitoring module plus 1 minute margin. As soon as a remote module should send a new data trame and ReMeS doesn't receive any, ReMeS can conclude that there is a problem with that connection.
    - If many remote modules aren't sending data, ReMeS will detect this later as a failure in the external communication channel. Suppose T is the transmission frequency of a module, we will detect this problem within $2 * T$ time.
  - Notification time
    - When only one remote module isn't sending any data, the owner of the module will be notified. Since this notification can happen through SMS or through email, there are different notification times. Through SMS the maximum time should be 5 minutes. If the owner is notified through email, the email will be delivered within 5 minutes, but we cannot predict when the owner shall read that notification email.
    - If more remote modules aren't sending data, we can conclude that there is a problem with the external communication channel. A system administrator will be notified about this within 5 minutes. He/she will then contact the telecom company in order to resolve the issue.
  - Resolution time
    - In the case that the external communication channel has a problem, we cannot ensure a fast recover of this channel. The SLA we have with the telecom company tells us that if there is a problem, it should be resolved within maximum 30 minutes.
    - When there is a problem with the signal between the remote module and the external communication channel, the time to resolve this issue depends on the client, the ReMeS operator and the ReMeS technician. If the client cannot resolve the problem within 2 days, a ReMeS operator will call the client to help him/her. If the ReMeS operator and the client are unable to resolve the problem, a ReMeS technician will be dispatched. This technician will go on scene to resolve the issue. Since the ReMeS technician will arrive within 5 days. Therefor the maximum repair time is 7 days.

6.1.2 Av2: Availability of the webportal

Because of a failure of our servers or because of a failure of the hosting service ReMeS uses, it is possible that the webportal where users can access their own information is not available all the time. There can also be a failure in the communication channel between the hosting service and the user. This communication channel is typically managed by an internet provider.

- **Source:** internal or external. If ReMeS rents a hosting service, the source is external. If ReMeS manages it's own servers, the source will be internal. In case of a combination, the source will be internal and external.
- **Stimulus:**
  - There is a failure in our servers or in the hosting service and nobody can access the webportal.
  - There is a failure in the communication channel provided by our (or the webhosting's) internet provider.
  - There is a failure in the communication channel provided by the user's internet provider. We will not consider this case since ReMeS has no SLA with this provider.
- **Artifact:** our servers or the servers of a webhosting service, external communication channel provided by an internet provider.

- **Environment:** normal execution
- **Response:**
  - Prevention:
    * In case that ReMeS rents a webhosting service, the ReMeS company has negotiated a Service-Level Agreement (SLA) with the webhosting service. This SLA states that the webservers are available in 99% of the time. They also ensure an average response time of 100 milliseconds.
    * In case that ReMeS manages its own servers, ReMeS hires technicians who maintain and improve the servers so they won't crash frequently. Also ReMeS will have multiple servers so that in case 1 of the servers crashes, another one will take over its jobs.
    * The ReMeS company has a SLA with the internet provider ensuring maximal availability of the external communication channel. The SLA ensures that the internet connection is available in 99% of the time. They also ensure an average response time of 80 milliseconds.
  - Detection:
    * In many cases the downtimes of the server are announced. These downtimes are needed for maintenance or other upgrades.
    * If a system goes down, it will typically send a notification to subscribed watchdog systems. These watchdog systems will notify the server administrator about the problem.
    * It is very unlikely that all components of the system crash at the same time (without sending a notification to subscribed watchdog systems). When one component crashes, another component that uses this one will notice that there is no longer a connection possible to this component. It will then notify the server administrator about the problem.
    * In the case that all components crash at the same time (without sending a notification to subscribed watchdog systems), the webportal of our ReMeS operators will also fail. The ReMeS operators will notify the server administrator in this case.
  - Resolution: In any case, the server administrator gets notified of the problem. The server administrator (and eventually other technicians) will try to reboot the servers and locate the error. In case that the servers won't reboot, they will have to locate and repare the error before rebooting the servers.
- **Response measure:**
  - Detection time
    * When all components crash at the same time without sending a notification to subscribed watchdog systems, a ReMeS operator will notice this rather fast, unless there is no ReMeS operator working at that time (ie at night). Therefor the maximum time for a detection of the problem will be 12 hours. We remind that the chance that this case happens is very very slim.
    * In any other case, the watchdog system or another component will detect the loss of connection very fast. The maximum time for detection of the problem will be 30 seconds.
  - Notification time Notifying the server administrator about the problem when the problem gets detected will take maximum 5 minutes. The server administrator must be available every time of every day, even at night. If the server administrator is unavailable for some period of time, he/she must provide a new contact person with atleast the same knowledge about the system.

- Resolution time
  - If the server administrator (and eventually other technicians) can reboot the servers, the resolution time will be maximum 60 minutes (providing time for the administrator to get to the location of the servers). Locating and reparing the error will take more time, but as long as the servers keep running, this isn't a problem.
  - In the case that rebooting the server is impossible, the downtime could take up to 24 hours. Remember that not necessarily all servers are down. It is very more likely that some servers are still up and running who can take over the job of the crashed server(s). In that case the users won't even notice the crashed server.

6.1.3 Av3: Consumption prediction model updates

The consumption prediction model that ReMeS creates for the utility companies is send periodically to these companies. Updates on the model can also be requested by the companies. ReMeS will send these updates within reasonable time.

- **Source:** internal and external
- **Stimulus:**
  - ReMeS will send these models periodically to the companies. It is time to send a new update.
  - A newer, better version of the model is available. ReMeS will send this model as fast as it can
  - A utility company requests a certain model. ReMeS will send this model as fast as possible
- **Artifact:** ReMeS Utility Production Planning
- **Environment:** normal execution
- **Response:** In each case ReMeS will send the latest available model. If ReMeS Utility Production Planning has almost finished calculating a newer version of the model, we will wait until this newer version is calculated and send this newer version to the utility companies.
- **Response measure:**
  - In the case that it is time to send a periodical update of the model, ReMeS will wait for the newest model only if the calculation will be finished within 30 minutes. The average time between two periodical updates should be around 24 hours.
  - If a newer, better version of the model is available, ReMeS will try to send this model as fast as possible. The delivery time depends solely upon the delay of the communication channel ReMeS uses to communicate with the utility company.
  - If a utility company requests a certain model (yearly, monthly, weekly, ...) ReMeS Utility Production Planning first has to compose this model. This should take 30 minutes at max. If the utility company requests the newest available daily model, ReMeS will send the latest model it has calculated unless the calculation of a newer model will be finished within 30 minutes.

6.2 Performance

6.2.1 PF1: Remote monitoring module sends new data

The remote monitoring modules will send data about the amount of consumed utilities, the battery state, signal level, leak detection, etc. ReMeS needs to respond correctly to this information.

- **Source:** internal
- **Stimulus:** A remote monitoring module sends a new trame of data containing valuable information.
- **Artifact:** System services
- **Environment:** Normal mode
- **Response:**
  - The trame contains data about the amount of consumed utilities. ReMeS will update its database so that the database has more correct values.
  - The trame contains a byte indicating that the battery is running low. ReMeS will notify a ReMeS operator and will also notify the client about the battery status.
  - The trame contains a byte indicating that the signal of the remote module connection is low. ReMeS will notify the client and a ReMeS operator about this alarm.
  - The trame's data indicates that there is a leak detected by the monitoring module. ReMeS will control the remote valve if available. ReMeS will notify the client about the leak and in the case of a gas leak, ReMeS will also notify the emergency services for gas.
- **Response measure:**
  - If the trame indicates that there is a leak, this alarm will receive the highest priority. ReMeS will notify the user within 5 minutes. If the leak is a gasleak, ReMeS will notify the emergency services within maximum 2 minutes. The remote valve (if installed) will also be controlled within maximum 2 minutes.
  - The other trames are receiving a lower priority. They will be handled within maximum 30 minutes.

6.2.2 PF2: A user requests data through the web portal

The web portal allows users to view their own consumption data. It also allows users to view personalized tips on how to lower their utility consumption. We assume that the web portal is available (see availability of the web portal)

- **Source:** external
- **Stimulus:**
  - The user logs in into the web portal
  - The user requests his/her measurement data over the last weeks, months or years
  - The user edits his/her contact information
  - The user edits the configuration of the remote monitoring module
  - The user requests consumption advice
- **Artifact:** System services
- **Environment:** Normal mode
- **Response:**
  - If the user logs in into the web portal, ReMeS will check if the username and password are consistent. If they are, ReMeS will provide a token for this user and the user will have access to his/her services.
  - If the user edits his/her contact information, ReMeS will update its database accordingly.

  - In the case that the user edits the configuration of the remote monitoring module, ReMeS will update this information in its database. ReMeS will also send a configuration trame to the remote monitoring module.
  - When the user requests measurement data or consumption advice, ReMeS will obtain the data from its database, process this data and show it to the user.
- **Response measure:**
  - Logging in into the web portal is a fairly easy task for ReMeS. From the moment the request is sent until the user sees the logged in information shouldn't take more than 30 seconds. In peak times, where the servers are overloaded, this could take longer.
  - Updating contact information also is an easy task. This operation shouldn't, as logging in, take more than 30 seconds.
  - Requesting and processing consumer data for one user is a slightly harder task. Therefor this could take almost 60 seconds until the user receives the processed data.
  - If the user wishes to edit the configuration of the remote monitoring module, ReMeS will send a configuration trame to the module. From the moment the user indicated the changes until the moment where the remote monitoring module changes its configuration could take up to 5 minutes. This because of the possible delay on the external communication channel.

6.2.3 PF3: A new consumer has been added to ReMeS

Whenever the utility companies that are a client of ReMeS gain new customers, these customers also needs to be added to our ReMeS system. In order to do this, ReMeS needs to gain access to the client's information to create a profile. Then ReMeS will create a username and password for this client. Finally the client will be able to log in to the ReMeS web portal with his authentication token.

- **Source:** external
- **Stimulus:** A client of a utility company calls the ReMeS call center with the request to become a client of ReMeS.
- **Artifact:** System services
- **Environment:** Normal mode
- **Response:**
  - When the client calls the ReMeS call center, ReMeS will send a contract to the client's home address. The client will fill in this contract and send it back to ReMeS.
  - When ReMeS receives the filled in contract, a new profile will be created in the ReMeS database. ReMeS will also generate an authorization token, consisting of a username and a password. This authorization token will be sent to the client. From now on, the client can log in using this data.
- **Response measure:**
  - The time between the moment that the client calls the ReMeS call center until ReMeS receives the filled in contract can be up to 5 working days plus the time that the client takes to send back the filled in contract.
  - Once ReMeS has received the filled in contract, it should only take maximum 1 day to activate the profile. Ofcourse the client will only receive the letter containing the authorization token 2-3 days later, depending on how fast the mail delivery service works.

7 Attribute Driven Design (ADD)

7.1 Iteration 1: Level 0

7.1.1 ReMeS

In the first iteration of the ADD runs, the full ReMeS system was used as a component to decompose. For this decomposition, several architectural drivers were chosen.

Architectural drivers

The primary decomposition for the ReMeS system started from certain key architectural drivers. The quality attributes chosen as key architectural drivers were the performance QAS and some Availability QAS. These chosen QAS also related to some functional requirements. To ease the documentation of the ADD runs, the related Use Cases are also mentioned. The architectural drivers that were chosen for this decomposition are:

- **P1: Timely closure of valves**
  - UC7: Send trame to remote device
  - UC13: Send alarm

  Performance 1 has certain qualities it demands. All the demands are about how certain alarm trames need to be processed. In case of a gas alarm trame, the incoming alarm trame must be processed with the highest priority. The outgoing control trame to close the gas valve also must be treated with the highest possible priority. This to ensure a response measure of less than 10 seconds between the alarm trame arriving and the control trame being sent. In case of a water alarm trame, the incoming alarm trame must be processed with normal priority. The outgoing control trame must be treated with the highest priority, like the outgoing control trames to close gas valves. The response time between an incoming water alarm trame and the corresponding outgoing control trame must be less than two minutes. Finally, in case of a power alarm trame, ReMeS must process incoming and outgoing with normal priority. The response time must be less than 3 minutes. We must also ensure a mechanism that avoids starvation of trame processing jobs.

- **P2: Anomaly detection**
  - UC10: Detect anomaly
  - UC9: Notify customer

  Incoming measurement trames will be processed by ReMeS to detect anomalies. It is possible that ReMeS receives new measurement trames at a faster rate than ReMeS can process them. In this case, ReMeS goes into overload modus. This modus will be activated when the throughput is greater than 50 anomaly detections per minute. In normal modus, ReMeS will process the incoming measurements in first-in, first-out order (FIFO). When ReMeS goes into overload modus, customers subscribed to the premium service level will get priority over the normal service level. Even while in overload modus, there should be no starvation of jobs and 98 percent of the measurements should be handled within 10 minutes after arriving. The load is also balanced over multiple instances of each sub-system.

- **AV1: Measurement database failure**
  - UC8: Send measurement

  It is possible that the internal database for storing measurements fails. This does not affect the availability of other types of persistent data. The database should be at least 99.9 percent be up and running. If it fails, detection of the failure should happen within 5 seconds and the operators are notified within 1 minute. In case of a failure, the database goes in degraded

modus. This means that incoming measurements are temporarily stored elsewhere and are processed when the database returns operational. This buffer should be able to store at least 3 hours of measurement trames. No measurements are lost when ReMeS switches from normal to degraded modus. The user interface should also display clearly that the measurement data is temporarily unavailable.

- **AV2: Missing measurements**
  - UC8: Send measurement

  It is possible that certain measurements are missing because of a external communication failure, a remote module failure or an internal subsystem failure. To detect this, ReMeS must be able to detect a single missing measurement. Remote devices and ReMeS must acknowledge received trames so they can detect failed trames. ReMeS should be able to detect the failure of an internal subsystem autonomously within 1 minute.

- **P3: Requests to the measurement database** In normal modus, the database processes the incoming requests first-in, first-out. Measurements for premium service level are handled within 500ms, other measurements are handled within 1500ms. If the system fails to comply to these deadlines, it goes into overload modus. This means that requests are handled in the order that returns the system to normal modus the fastest, keeping in mind that requests from premium service level customers are handled before other requests and history queries for anomaly detection are prioritized over research purpose history queries. In In overload modus, consumer history queries are allowed to return a stale cached version.

- **M1: Dynamic Pricing**
  - UC15: Generate invoice

  In the near future, utility prices will change dynamically on a minute-to-minute basis. Since ReMeS already has the necessary infrastructure in place to remotely communicate with many customers, providing the current price to them seems like a logical extention. ReMeS will ensure that this modification takes less than 250 man man months to implement. The costs of this implementation will cost less than 2Mio Euro.

Tactics

To address these drivers, certain tactics have been used. Mainly, performance tactics have been used.

- Performance
  - Resource management
    For considering resource management, the introduction of concurrency and the increase of available resources are the chosen tactics. Increasing resource management is, however, out of the scope of these add runs.
  - Resource arbitration
    For considering resource arbitration, different scheduling policies can be used.
- Availability
  - Fault Detection
    The tactic that was considered to address the most important drivers (for this level of decomposition), was the use of a heartbeat mechanism. The eventual decision and explanation will be given later on.

Architectural Patterns

The architectural patterns used to effect the tactics chosen in the previous section will be explained in this section.

- Active Object
  Concurrency is an important requisite for the system that is being developed. Being able to execute operations of components within their own threads of control can help achieve this goal. The use of an active object architecture also improves the ability to issue requests on components without blocking until the requests execute.

  It should also be possible to schedule the execution of requests according to specific criteria, such as customer type based priorities and load based priorities.

- Command processor
  The command processor pattern will be used for the implementation of different schedulers. Often the functionality of the scheduler, as described in the command processor pattern, will be split up into two separate components. These components are the buffer where the service requests are stored and the actual scheduler that empties the buffer and schedules the commands for execution. This shows a more explicit view of the workings of such a scheduler without going into another deeper level of ADD design.

- Server Request Handler
  To be able to easily receive messages from the remote modules, we opted for a server request handler pattern.

  It is also possible to use strategies in the aforementioned schedulers. But this will be explained in further iterations of the ADD process when needed.

- Heartbeat
  In order to conform to the driver stating that the system should know about failing components, a heartbeat mechanism will be implemented. Every component that is important enough to be registered, sends heartbeats to the watchdog component at frequent intervals. When the first heartbeat arrives, the watchdog keeps track of that components uptime and notifies the event channel if certain heartbeats are missing. This way components can be dynamically added to the systems uptime monitor (or in this case watchdog).

  The choice was made not to use ping/echo because that would assume a one to many relationship of dependency between the Watchdog and the outside components. In terms of scalability that would lead to a poorer design than the many to one relationship that is in place by using heartbeats. By using heartbeats, the watchdog does not need any knowledge of the system layout or structure in order to function. It is almost completely passive in it's functionality. The responsibility for deciding on the importance of failure detection lies completely with the other components.

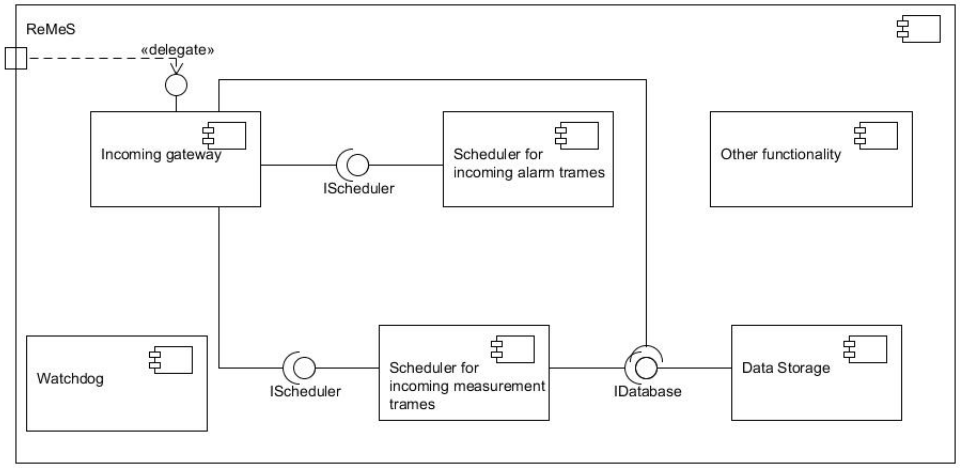

Figure 3: The ReMeS component diagram first decomposition.

The result of this iteration of the ADD process can be shown in figure 3

Verification and refinement of drivers

After the initial decomposition no quality attributes have been completed yet. All considered drivers will be delegated to child components to act as drivers for their individual decomposition.

Incoming gateway

- **UCx:** Know the type of the trame
- **UCz:** Send acknowledgement
- **UC8':** Send Measurement
- **UC13':** Send alarm
- **P1':** Timely closure of valves

Scheduler for incoming measurement trames

- **UC8':** Send measurement
- **AV1:** Measurement database failure
- **AV2:** Missing measurements
- **P2:** Anomaly detection
- **UCy:** Know the modus
- **UCw:** Retrieve module checking schedule

Scheduler for incoming alarm trames

- **UC13':** Send alarm
- **P1':** Timely closure of valves

Data storage

- **UC8':** Send measurement
- **UC10:** Detect anomaly
- **AV1:** Measurement database failure
- **P2:** Anomaly detection
- **P3:** Requests to the measurement database
- **UCy:** Know the modus

Watchdog

- **AV1:** Measurement database failure
- **AV2:** Missing measurements

Other functionality

- **P1':** Timely closure of valves
- **UC7:** Send trame to remote device
- **UC13':** Send alarm
- **UC9:** Notify customer
- **M1:** Dynamic pricing
- **UC15:** Generate invoice

- **UC16:** Mark invoice paid

7.2 Iteration 1: Level 1

7.2.1 Incoming gateway

In this section, we will decompose the "Incoming Gateway" component.

Architectural drivers

The main purpose of the incoming gateway is separating the measurement from the alarm trames and other types of trames in the future, if need be. Therefore the chosen architectural drivers are the following:

- **UCx: Know the type of the trame**
  This use case is a new use case. It is not described in the assignment. What this use case will do is decompose the incoming trame to find out what type of trame it is.
- **UCz: Send acknowledgement**
  This use case is a new use case. If a measurement or alarm trame is received by ReMeS, ReMeS will send an acknowledgement of receiving the trame back to the sending device.
- **UC8': Send measurement**
  The incoming gateway will partially solve the use case "Send measurement" because it will make sure that the measurement trame is sent to the correct component of the system, after receiving the measurement trame and acknowledging it.
- **UC13': Send alarm**
  The incoming gateway will partially solve the use case "Send alarm" because it will make sure that the alarm trame is sent to the correct component of the system, after receiving the alarm trame and acknowledging it.
- **P1': Timely closure of valves**
  A mechanism should be in place to avoid starvation of (alarm) trame processing jobs. Alarm trames should be handled as soon as possible and ultimately within a hard deadline.

Tactics

To efficiently solve these drivers, the following tactics were used.

- Performance
  - Reduce computational overhead
    In order to maximize the performance of the system, it is important to minimize the computational overhead caused by switching between functionality based on the type of incoming trame.

Architectural Patterns

The architectural patterns that are used to effect the tactics are the following:

- Message Router
  The message router pattern localizes the router logic for the incoming trames. By using a message router implementation in the incoming gateway, it is possible to cleanly separate different functionalities from each other. It is also has a high impact on modifiability aspects of the system. Although this is not one of the main QAS, it is considered an important factor in good quality architectures to be extensible and minimize the ripple effect of adding new

changes. By using this pattern, adding new functionality based on new types of trames is made easier.

Message routers consume messages from one message channel (the channel where all external messages are received) and reinsert them into different message channels, depending on a set of conditions (alarm or measurement trames).

- Invoker
  To be able to easily receive messages from the remote modules, we opted for an invoker pattern.

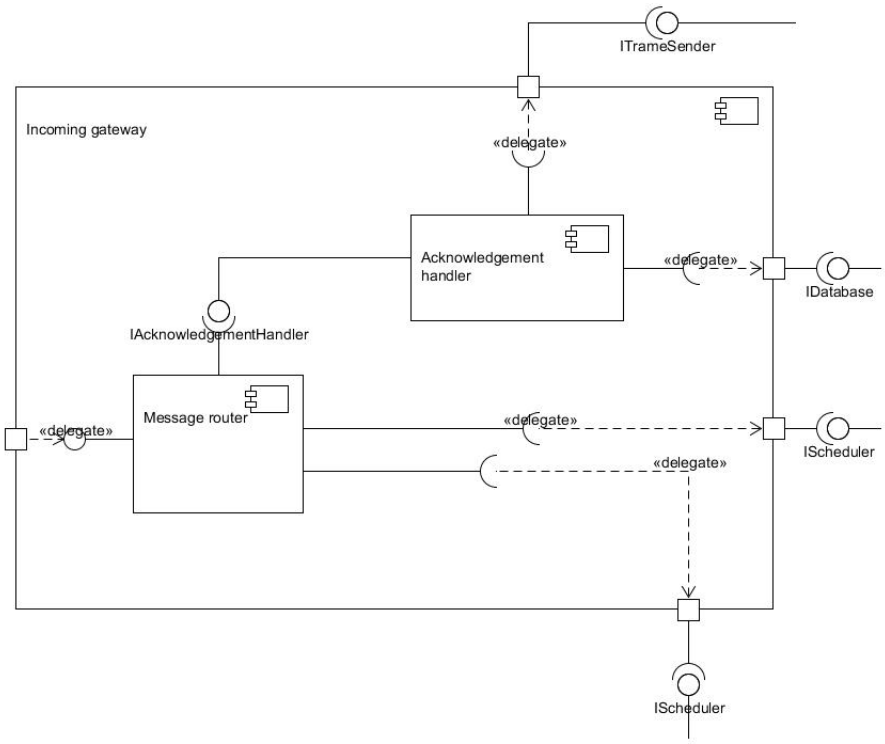

Figure 4: The Incoming Gateway after decomposition

Verification and refinement of drivers

After this decomposition, UCx (Know the type of trame) will be satisfied by the Message Router component. The partial functionalities and quality attributes described in UC8'(Send measurement), UC13'(Send alarm) and P1'(Timely closure of valves) are all handled and satisfied in the Message Router component. The sending of acknowledgements on receiving certain trames, which is described in UCz(Send Acknowledgement) is handled by the component Acknowledgement Handler.

Message Router

- **UCx:** Know the type of trame
- **UC8':** Send measurement
- **UC13':** Send alarm
- **P1':** Timely closure of valves

Acknowledgement Handler

- **UCz:** Send Acknowledgement

7.2.2 Scheduler for incoming measurement trames

This section will explain our reasons behind the decomposition of the "Scheduler for incoming measurement trames" component.

Architectural drivers

This component has to ensure that the incoming measurement trames are handed in the correct order to the data storage component. It also must temporary store the incoming measurement trames in case that the measurement database fails. Lastly, this component detects missing measurements.

- **UC8': Send measurement** If a trame from a remote device is the first device sent, the system will mark the device as active.
- **AV1: Measurement database failure** The scheduler for incoming measurement trames contains a buffer that can store at least 3 hours of data in case of a measurement database failure.
- **AV2: Missing measurements** ReMeS must be able to detect missing measurement updates from remote modules and notify an operator.
- **P2': Anomaly detection** In normal modes ReMeS processes the incoming measurements in a first-in, first-out order. In overload modus, ReMeS gives priority based on the SLA with the customer.
- **UCy: Know the modus** For the system to know what order to hand the measurement trames to the data storage component, it must know the current modus of the system.

Tactics

- Performance
  - Scheduling Policy
    To ensure the correct order or the incoming measurement trames, keeping in mind the modus of the system and the subscription type of the customers we chose for a scheduling policy

Architectural Patterns

- Active Object
  The active object pattern allows us to insert new measurement trames in a scheduler. The scheduler will then decide itself which trames should be processed first. The active object will be implemented as a command and will be handled by a command processor structure in this case. The buffer that is used by the scheduler should also be large enough to hold at least 3 hours of data in case of a measurement database failure. This way, the availability driver will be addressed.
- Publish-Subscribe
  This pattern allows us to efficiently notify all interested parties about the modus of the system and more importantly, to be notified of changes in modus caused by other components. We will also use this pattern to notify an operator through the user interface by this publisher-subscriber pattern. The reason for notifying operators will be addressed in a further decomposition.

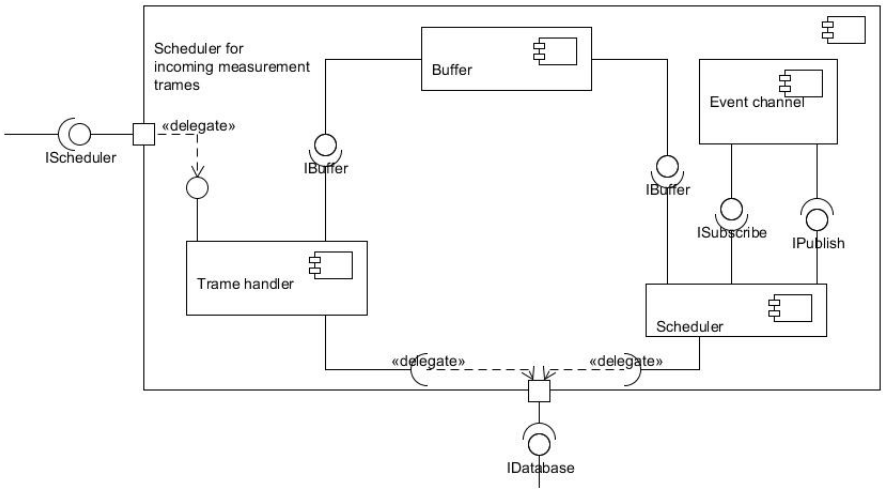

Figure 5: The Scheduler for Incoming Measurement Trames after decomposition

Verification and refinement of drivers

After the decomposition, UCy and P2 are fully satisfied by the scheduler. AV1 will be partially handled by the buffer of the scheduler component. This buffer will be large enough to contain 3 hours worth of data. AV1 and AV2 will be (at least partially) delegated to child nodes Trame Handler and Buffer for further decomposition. The Decomposition of Scheduler could show the implementation of a strategy pattern to effect the different policies of scheduling.

Trame Handler

- **AV2: Missing measurements**

Buffer

- **AV1: Measurement database failure**

7.2.3 Scheduler for incoming alarm trames

This section will explain our reasons behind the decomposition of the "Scheduler for incoming alarm trames" component.

Architectural drivers

The main purpose of this component is making sure that the trames are processed in the correct order, according to their priority. Incoming gas alarm trames must be treated with the highest priority and the incoming water and power alarm trames with normal priority. This component will make sure that this happens. This component will handle the following drivers:

- **UC13': Send alarm** The scheduler for incoming alarm trames will partially help in solving this use case since it will notify the correct system component to send a control trame to the remote valve.
- **P1': Timely closure of valves** ReMeS will process incoming gas alarm trames with a higher priority as water and power alarm trames. ReMeS must also make sure that starvation of alarm trames is not possible.

Tactics

- Performance
  - Scheduling Policy
    To ensure that certain incoming alarm trames such as gas alarm trames are treated with the highest priority, we chose for a scheduling policy. The scheduling policy will also make sure that starvation is not possible.

Architectural Patterns

- Active Object
  The active object pattern allows us, as in the Scheduler for incoming measurement trames, to insert new alarm trames in a scheduler. The scheduler will then decide itself which trames need to be processed first. The active object will be implemented in the same way as in the Scheduler for incoming measurement trames. The buffer that is used by the scheduler shouldn't be as big as with measurement trames.
- Proxy
  To increase the modifiability of the internal working of this component we chose to use the explicit interface pattern. The interface that we see outside the component will simply delegate it's methods to an internal component.

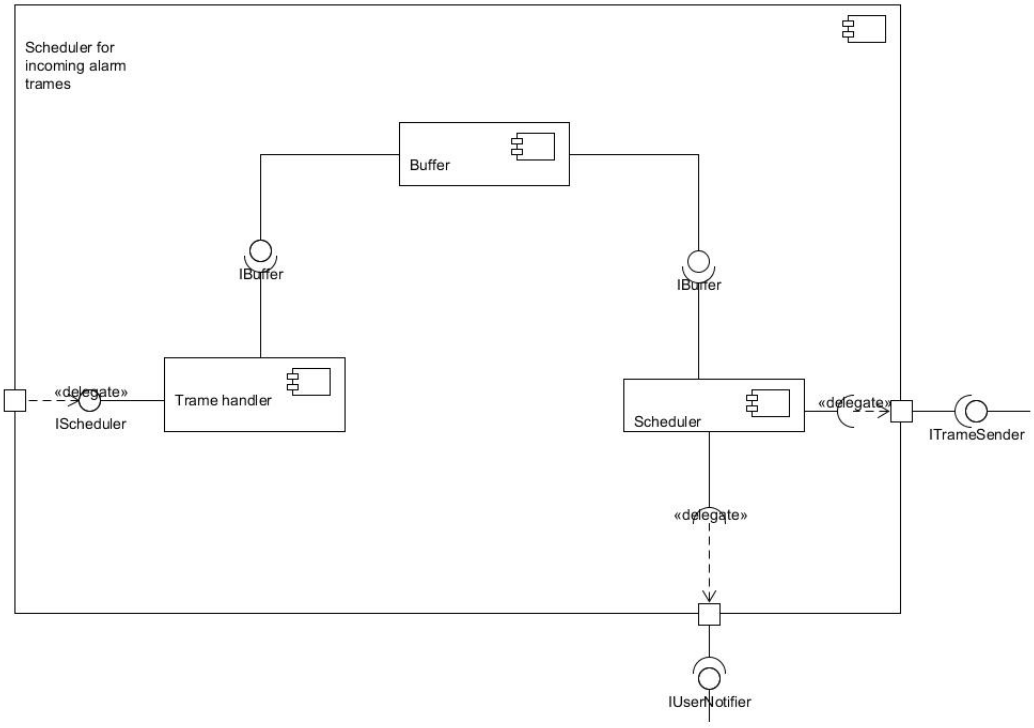

Figure 6: The Scheduler for Incoming Alarm Trames after decomposition

Verification and refinement of drivers

After the decomposition, P1 will be partially satisfied by the scheduler. Some incoming alarm trames will be prioritized while starvation of certain incoming alarm trames is not possible. UC13 will be delegated to another component in our system.

7.2.4 Data Storage

In this section, the data storage component will be decomposed.

Architectural drivers

- **UC8':** Send measurement
  ReMeS looks up the customer associated with the device sending the measurement, processes and stores the measurement in his consumption record.
- **UC10:** Detect anomaly
  ReMeS checks measurements for anomalies. If anomalies are found, the appropriate recipients need to be notified and possibly actuators need to be operated remotely. For the detection of anomalies, also all of the measurement history needs to be fetched from the database.
- **AV1:** Measurement database failure
  If the database fails, it should do so gracefully and not impact other services for the time being. All concerning components need to be notified when going down.
- **P2:** Anomaly Detection
  There are different modi to be considered: A normal mode and an overload mode. Multiple instances need to handle the load that has to be processed for load balancing purposes. All requests need to be handled eventually. No starvation can occur while processing requests.
- **P3:** Requests to the measurement database
  In normal modus, the database processes the incoming requests first-in, first-out. If the system fails to comply to the specific deadlines listed below, it goes into overload modus in which requests are handled in the order that returns the system to normal modus the fastest, thereby taking into account the SLA with the customer and the origin of the request. It should be possible to return a stale cached version when in overloaded modus.
- **UCy:** Know the modus
  As previously indicated this module needs to be aware of the modus of operation, and possibly be able to change that modus.

Tactics

- Performance
  - Resource arbitration
    A scheduling policy will need to be applied in order to address the case of scheduling requests while keeping in mind the different priorities and origins of the requests.
  - Resource Management
    In order to address the case of load balancing, at least some form of concurrency needs to be effected. These last two tactics take precedent over any other tactics that may conflict because of the priority assigned to the quality attributes from which these tactics where chosen.
- Availability
  - Fault detection
    In order to address the graceful failure of different components in the system, said failure needs to be detected before it can be addressed. This however will be done by the watchdog component described in higher levels. It is merely mentioned here as a tactic because this component will probably be heavily influenced.

Architectural Patterns

- Active object
  In order to implement and use the different tactics to decompose this module, certain patterns were (re)used. This architecture relies quite heavily on the active object pattern. These active objects will, again, be implemented as commands which will be scheduled by a command processor. The choice of reusing the active object implementation makes it possible to pass along commands as service requests in a more easy and general way, improving performance (not needed to convert messages all the time) and improving extendability.
- Publish/Subscribe
  The publish/Subscribe (or Pub/Sub) pattern is reused to provide components of a mechanism to know and if needed, let other components know of a change in modus. For ease of explanation, the event channel is shown as a new component, but actually, this component is the same component as shown in other decompositions. Each component in the ReMeS system that uses an event channel, actually uses the same event channel.

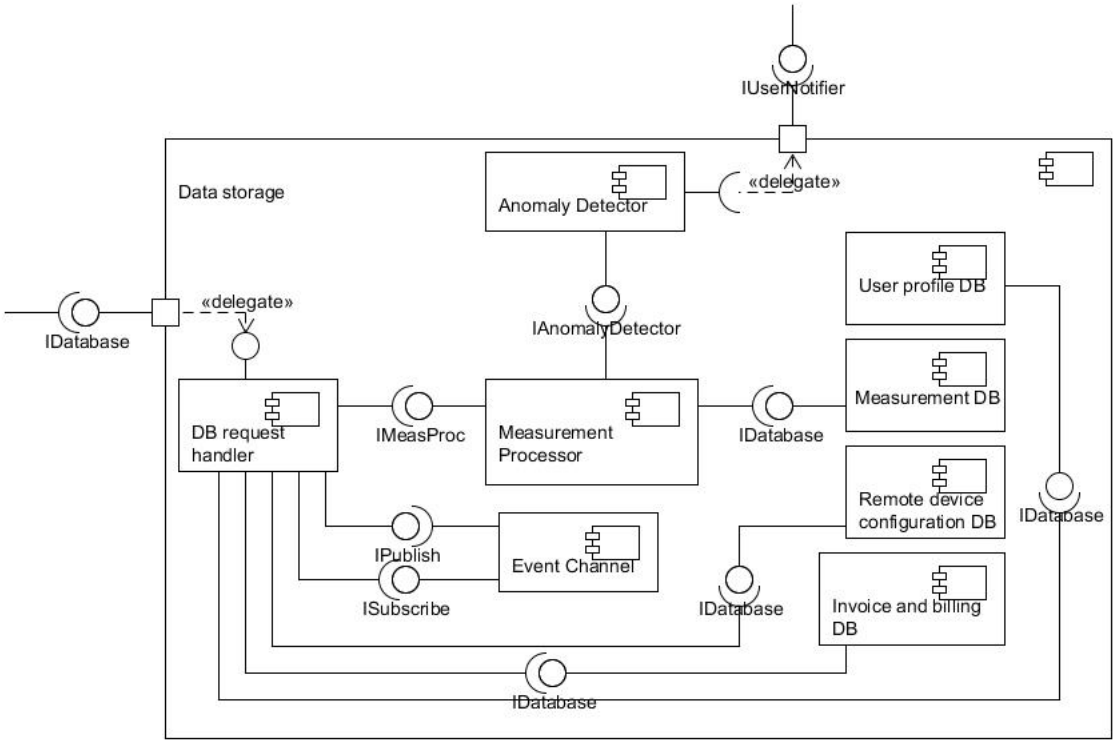

Figure 7: The Data Storage component after decomposition

Verification and refinement of drivers

For this decomposition, UC8(Send Measurement) has been completely addressed. UC10(Detect anomaly) has been addressed but the functionality of notifying alarm recipients has been delegated outside this component. UCy has also been completely addressed. As for the detection of failures as described in Av1, this functionality has been delegated to an outside component (watchdog). Only P2 and P3 (the major drivers) are only partially addressed and need to be explained further in a lower level decomposition although even in this level the structures for addressing these drivers have already been implemented.

DB Request Handler

- **P3:** Requests to the measurement database

Anomaly detector

- **P2:** Anomaly detection
- **P3:** Requests to the measurement database

7.2.5 Watchdog

Architectural drivers

The main drivers for this component are the quality attributes that are related to the detection of failing components. These events should be detected and operators should be notified within a certain time interval. In this case the time interval is 5 seconds.

- **AV1:** Measurement database failure The detection of failed components happens within 5 seconds.
- **AV2:** Missing measurements The failure of any component in the system should be detected and an operator should be notified.

Tactics

The tactics used to effect these drivers are the following.

- Availability
  - Failure detection
    The use of a heartbeat system as described in the upper layer decomposition of the ReMeS system will be used.

Architectural Patterns

- Heartbeat
  Each component for which failures should be detected, should be instantiated with a reference to the watchdog component. Every component should send at least one heartbeat within a 5 second deadline of the former heartbeat sent.

  The watchdog component monitors these heartbeat and starts tracking a component from the moment a first heartbeat arrives. When the watchdog detects missing heartbeats, a notification is sent to the event channel for all interested parties to see.

  Although not one of the main drivers, the modifiability attribute will be affected in a positive manner because of this form of runtime registration. Deferring the binding time in this manner causes a lower coupling between this component and the others in the system.
- Publish/Subscribe
  The publish/subscribe pattern which is used in a few other decompositions all over the system, is also used in this component to notify interested parties of a change in state of a component. This change in state is determined by the absence of heartbeats. The event channel depicted, is the same event channel used throughout other components in the system.

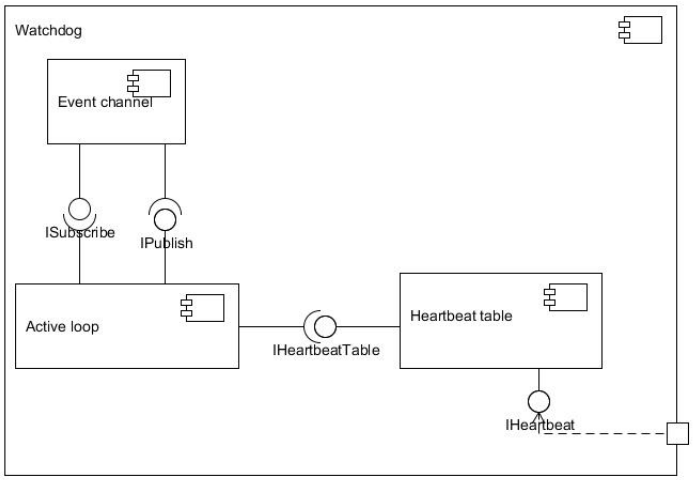

Figure 8: The Watchdog after decomposition

Verification and refinement of drivers

All drivers for this decomposition have been addressed. No further decomposition of this module is needed.

7.3 Iteration 1: Level 2

7.3.1 Trame Handler

Architectural drivers

- **UCw: Retrieve module checking schedule**
  For the scheduler for incoming measurement trames to check whether or not some measurements are missing, it must be able to retrieve the rate that the remote module will be sending measurements. It can then compare the timestamps of the latest measurement trames with this rate to see if some measurements are missing.
- **AV2: Missing measurements**
  ReMeS should be able to detect a single missing measurement by monitoring the measurement schedule Operator should be notified after missing measurements.

Tactics

- Availability
  - Fault detection
    An implementation of a Heartbeat system is used to check whether all devices send their measurements in time.

Architectural Patterns

- Heartbeat
  The device's measurement trames are used as heartbeats in this implementation. An active loop component keeps track of the new entries in the heartbeat table and monitors them for absent measurements. The active loop component also has the capability of querying the

data storage for the send schedule of the different devices. This component can also mark devices as active in the data storage component when a device sends it's first trame.

- Active Object
  The active object pattern is used to provide a way of making an object run independently in its own thread of control for monitoring the heartbeat table for missing measurements. The active loop component represents an active object in this decomposition.

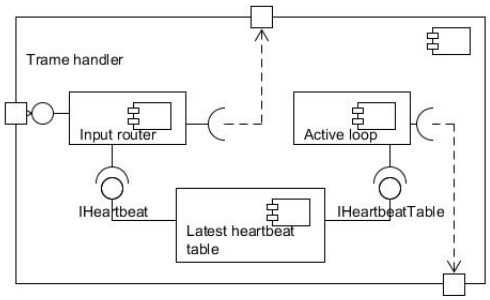

Figure 9: The Trame Handler after decomposition

Verification and refinement of drivers

All drivers for this module have been met and addressed. No further delegation to child nodes is needed.

7.3.2 Buffer

This section handles the decomposition of the Buffer component inside the Scheduler for incoming measurement trames.

Architectural drivers

- **AV1: Measurement database failure**
  When switching between normal and degraded modus, no new measurements are lost. When the database component fails, incoming measurements should be stored elsewhere and it should be possible to store at least 3 hours of measurement messages.

Tactics

- Availability
  - Recovery Preparation and repair
    In order to make sure that 3 hours worth of messages can be stored in case something goes wrong, and to make this storage more prepared against failures, some form of active redundancy needs to be implemented.

Architectural Patterns

- Replicated Component group
  The replicated component group pattern ensures that the buffer holding the measurement trames is in fact distributed over different locations making this buffer more redundant. In

case of failure no new measurements will be lost since there will always be a buffer instance ready to store this data. The buffer should be able to store 3 hours worth of data.

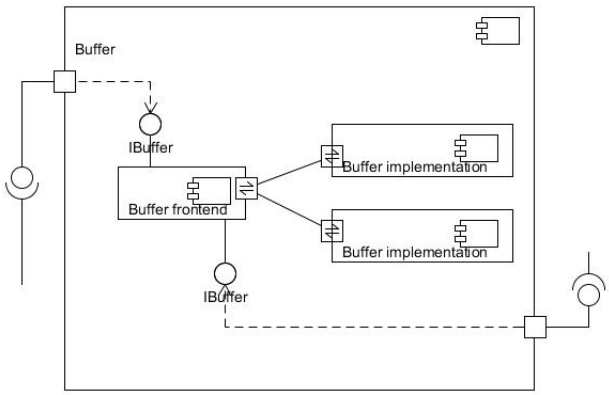

Figure 10: The Buffer after decomposition

Verification and refinement of drivers

All drivers for this module have been addressed and fulfilled.

7.3.3 DB Request Handler

This section handles the decomposition of the DB Request Handler inside the Data Storage component.

Architectural drivers

- **P2:** Requests to the measurement database
  In case of a database failure, a stale cache version should be returned when requests can't be handled.

Tactics

- Performance
  - Resource Management
    In order to conform to the drivers, a form of cache should be used to maintain multiple copies of data.
- Modifiability
  - Localize Changes
    Since this module is the main entry point for requests to the data storage unit, it is important to offer a general interface to the outside components. Generalizing the module will address this.

Architectural Patterns

- Proxy
  A proxy is used to offer the generic data storage component interface to the other components in the system. This improves the simplicity in making requests to the database. This also unifies the different types of database services offered into one accessible interface.

- Active Object
  Again the active object is used and implemented as a command alongside a command processor structure. Data transfer objects will be used to transfer data to and from the databases itself.
- Chain of responsibility
  In order to accommodate for the caching of data, the chain of responsibility can be used to give the parallel processor and the Request Cache the opportunity to decide themselves if they can handle a request or if they should forward it. The Request Cache component can forward the request to the request process in cache a request is not available in cache and a request to the actual database needs to be made.

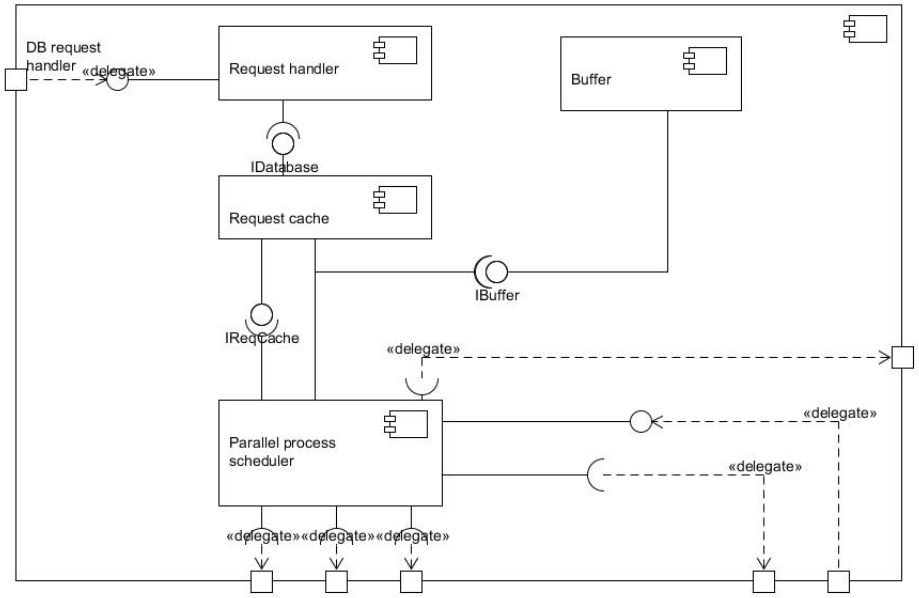

Figure 11: The DB Request Handler after decomposition

In the diagram of the DB request handler, the ports that are shown to connect this component to the outside components are placed in the same location as the connections of this component in a diagram representing a level higher. Such a higher level diagram can be found in figure 7. The triplet of outgoing ports connect to the different database instances and the duo of connectors to the side interface with the event channel.

Verification and refinement of drivers

Using these patterns, the drivers for this decomposition have been fully addressed. There is no real need to decompose this module further.

7.4 Iteration 2: Level 1

7.4.1 Other functionality

Decomposing this module actually represents doing another iteration of ADD. In further diagrams, the components that will arise from this decomposition will not be subcomponents of the other functionality module since this was only a pseudo module for ADD-process purposes.

Solely the drivers from the first decomposition that were delegated to this pseudo-component will be handled. If another iteration is needed for quality attributes or functionality that is not covered by these drivers, then another iteration will follow after this one.

Architectural drivers

- **P1': Timely closure of valves**
  - UC7: Send trame to remote device
  - UC13': Send alarm
  - UC9: Notify customer

  Performance 1 has only been fulfilled partially in the first iteration. Our current ReMeS decomposition is unable to send outgoing messages or trames. But to close the valves we obviously need outgoing control trames. These outgoing control trames also must be treated with different priorities. Again, starvation of certain outgoing messages cannot happen.
- **M1': Dynamic pricing**
  - UC15: Generate invoice

  In the near future, utility prices will change dynamically on a minute-to-minute basis. Since ReMeS already has the necessary infrastructure in place to remotely communicate with many customers, providing the current price to them seems like a logical extention. ReMeS will ensure that this modification takes less than 250 man man months to implement. The costs of this implementation will cost less than 2Mio Euro.

Tactics

- Performance
  - Resource arbitration
    For considering resource arbitration, different scheduling policies can be used.

Architectural Patterns

We will be using mainly the same patterns as we used in the first decomposition of ReMeS. This is because the motivation and the reasoning behind the first decomposition on this level hasn't changed.

- Active Object
  Concurrency is an important requisite for the system that is being developed. Being able to execute operations of components within their own threads of control can help achieve this goal. The use of an active object architecture also improves the ability to issue requests on components without blocking until the requests execute.

  It should also be possible to schedule the execution of requests according to specific criteria, such as customer type based priorities and load based priorities.
- Command processor
  The command processor pattern will be used for the implementation of different schedulers. Often the functionality of the scheduler, as described in the command processor pattern, will be split up into two separate components. These components are the buffer where the service requests are stored and the actual scheduler that empties the buffer and schedules the commands for execution. This shows a more explicit view of the workings of such a scheduler without going into another deeper level of ADD design.

- Client Request Handler
  To send messages across the internet, through sms or gprs, we opted for a client request handler pattern.

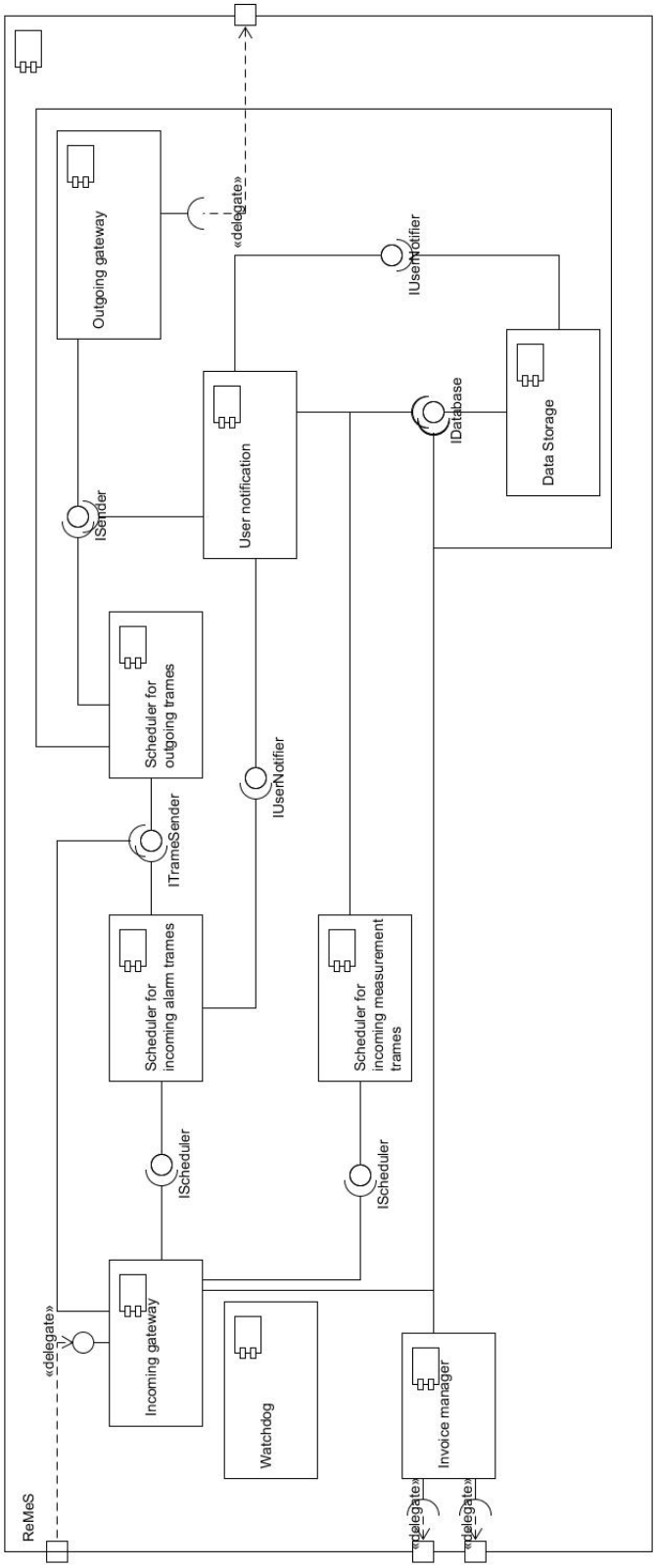

Figure 12: The decomposition of ReMeS after the second iteration

Verification and refinement of drivers

No quality attributes have been completed. All drivers are delegated to child components:

Scheduler for outgoing trames

- **P1':** Timely closure of valves
- **UC13':** Send alarm
- **UC7':** Send trame to remote device

User notification

- **UC9':** Notify customer
- **UC13':** Send alarm

Outgoing gateway

- **UC7':** Send trame to remote device
- **UC9':** Notify customer

Invoice Manager

- **M1:** Dynamic pricing
- **UC15:** Generate invoice

7.4.2 Scheduler for outgoing trames

Architectural drivers

This component will manage outgoing control trames. It will prioritize important trames over others without having starvation. This component also has to construct the outgoing control trames, so that the receiver can correctly read all the contained information. Finally, this component will determine the communication channel to reach the remote device with.

- **P1': Timely closure of valves** Outgoing valve control trames have a certain priority. Gas and Water valve control trames have the highest priority. Others have normal priority.
- **UC7': Send trame to remote device** ReMeS wants to send trames to remote devices to control those devices.
- **UC13': Send alarm** ReMeS has to send control trames to the remote actuators to close them in case that a leak is detected.

Tactics

- Performance
  - Scheduling Policy
    To ensure the correct order of outgoing trames to remote devices, we chose for a scheduling policy.

Architectural Patterns

- Message Translator
  To construct the trame to send to the remote device, we opted for a message translator pattern

- Active Object
  The active object pattern allows us to insert new outgoing trames (or atleast requests for sending outgoing trames) in a scheduler. The scheduler can then decide which requests to handle first.
- Proxy
  To increase the modifiability of the internal working of this component we chose to use the explicit interface pattern. The interface that we see outside the component will simply delegate it's methods to an internal component.

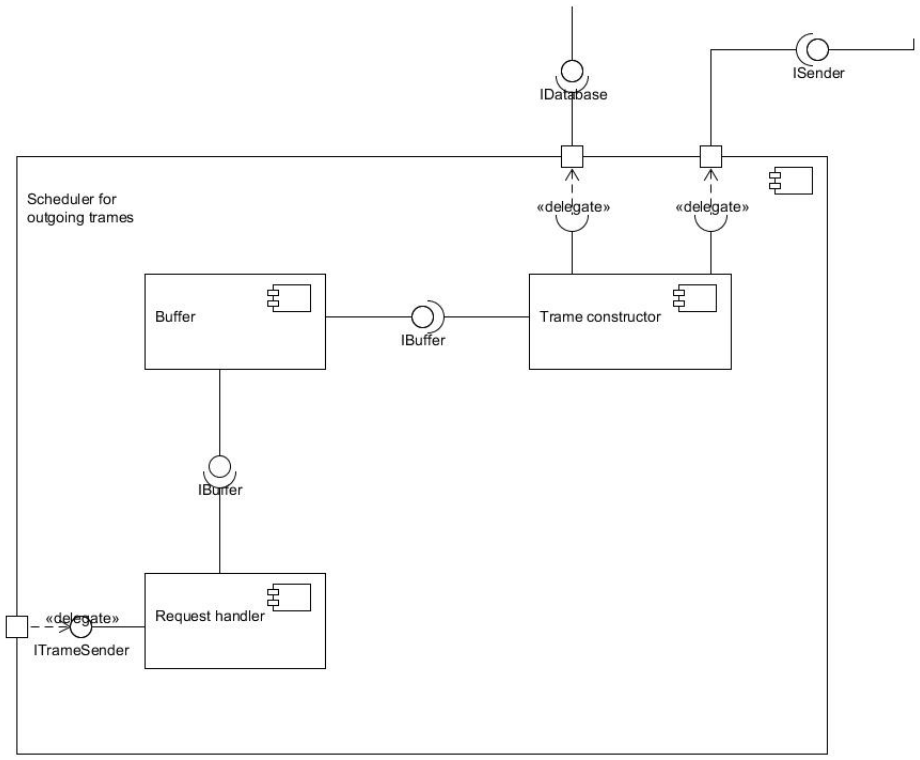

Figure 13: The Scheduler for Outgoing Trames after decomposition

Verification and refinement of drivers

The final part of P1 has been handled by this component. UC13' also was fulfilled by this component. Only UC7 is partially finished. The trames to send have been constructed, but not yet sent. But that part of UC7 will be done by another component in the system.

7.4.3 User notification

Architectural drivers

This component will manage outgoing customer notifications. It wont prioritize any requests. The component handles the request First-in-First-out (FIFO). The component will construct the messages and determine which communication channel to use.

- **UC9': Notify customer** Whenever ReMeS wants to send a notification to a customer, ReMeS should address this component.

- **UC13': Send alarm** When ReMeS receives an alarm trame, the customer needs to be notified (if the customer has indicated this in his profile).

Tactics

- Performance
  - Resource arbitration
    Different scheduling policies can be used. Currently the FIFO scheduling policy is used in this component.

Architectural Patterns

- Message Translator
  To construct the notification to send to the remote device, we opted for a message translator pattern. This pattern allows us to create standard user notifications such as "Gas leak detected in your home."
- Active Object
  The active object pattern allows us to insert new user notification requests in a scheduler. The scheduler can then decide which requests to handle first. Currently the scheduler will handle the requests FIFO.
- Proxy
  To increase the modifiability of the internal working of this component we chose to use the explicit interface pattern. The interface that we see outside the component will simply delegate it's methods to an internal component.

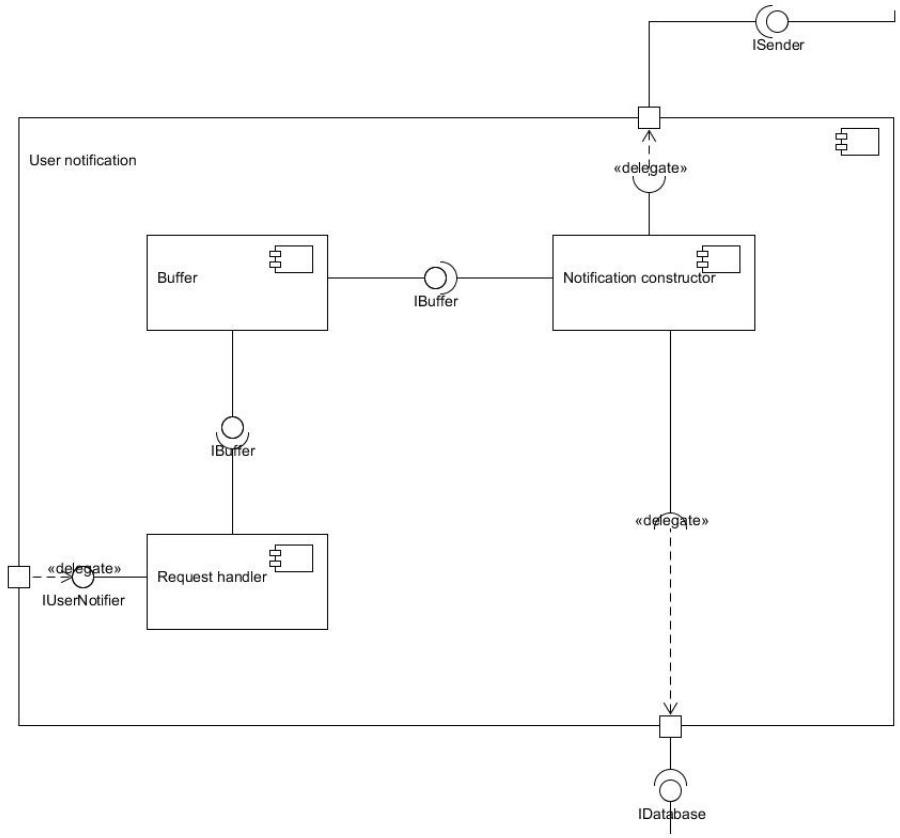

Figure 14: The User Notification component after decomposition

Verification and refinement of drivers

Both UC9' and UC13' have been handled by this component. The second part of UC9 (sending the message) will be handled by the outgoing gateway.

7.4.4 Outgoing gateway

Architectural drivers

This component can construct SMS, gprs or internet messages and send them.

- **UC7': Send trame to remote device** The final part of this usecase is sending the trame through the selected communication channel.
- **UC9': Notify customer** ReMeS will send the constructed message to the recipient.

Tactics

- Performance
  - Resource arbitration
    Different scheduling policies can be used. Currently the FIFO scheduling policy is used in this component.

Architectural Patterns

- Active Object
  The active object pattern allows us to insert new user notification requests in a scheduler. The scheduler can then decide which requests to handle first. Currently the scheduler will handle the requests FIFO.
- Proxy
  To increase the modifiability of the internal working of this component we chose to use the explicit interface pattern. The interface that we see outside the component will simply delegate it's methods to an internal component.
- Requestor
  To send messages across the internet, through sms or gprs, we opted for a requestor pattern.

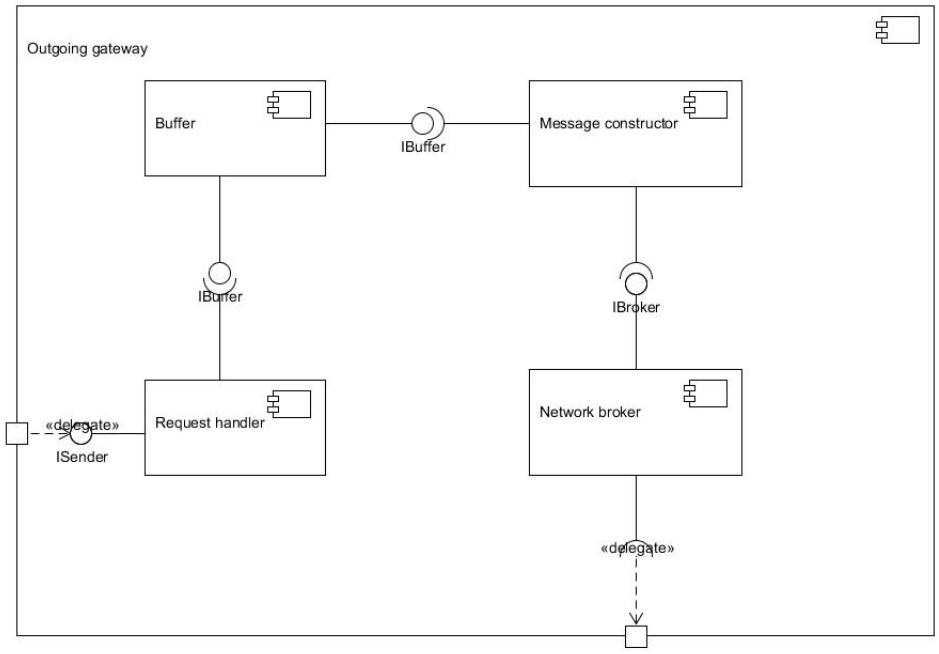

Figure 15: The Outgoing Gateway after decomposition

Verification and refinement of drivers

All the architectural drivers have been satisfied.

7.4.5 Invoice manager

Architectural drivers

This decomposition starts solely with a functional requirement

- **M1:** Dynamic pricing
  Room should be left for extending the functionality of the ReMeS system to incorporate a smart billing service. Communication between UIS and ReMeS should be made possible. Billing and invoicing generation and handling should be provided. All this does not affect remote metering, actuation, leak and anomaly detection.

- **UC15:** Generate invoice
  ReMeS processes the measurements since the last bill and constructs the invoice. ReMeS contacts the Utility Provider to get contract information and pricing information. The invoices are stored in ReMeS and marked as unpaid.

Tactics

- Modifiability
  - Defer Binding Time
  - Prevention of ripple effect

Architectural Patterns

- Explicit Interface
  The use of an explicit interface allows this component to change it's internal functionality and components independently from the interface it provides to the outside components.

  In combination with different ways of triggering the functionality, depending on the final implementation, the components that will be affected by a change, will be minimized. The different ways of triggering have been addressed in the ADD assumptions in section 7.6

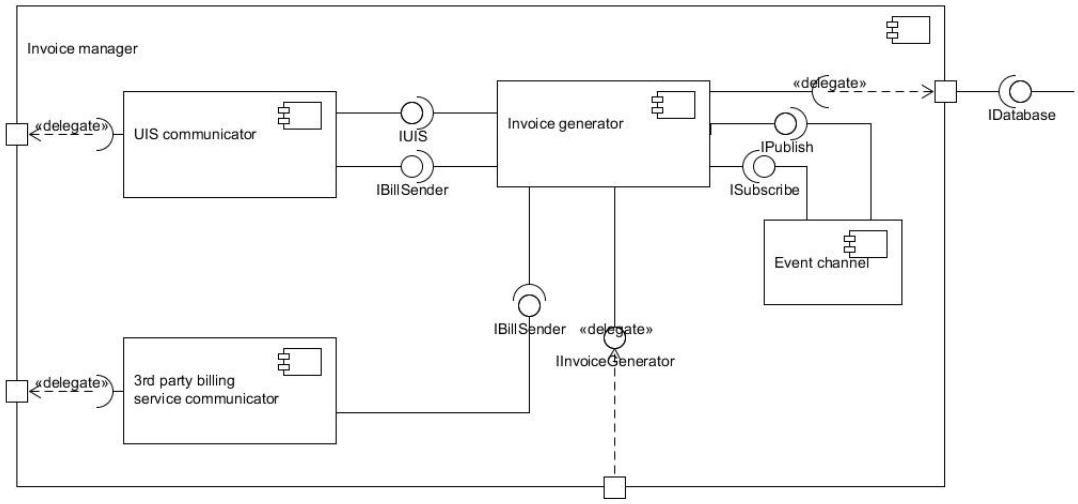

Figure 16: The Invoice Manager after decomposition

Verification and refinement of drivers

It is possible to refine some components further since not all functionality has been addressed yet.

Invoice generator

- **UC15:** Generate invoice

UIS communication handler

- **M1:** Dynamic pricing

7.5 Iteration 3: Level 1

7.5.1 ReMeS

This decomposition will manage all use cases that are not related to any QAS.

Architectural drivers

- **UC1:** Log in
- **UC2:** Log off
- **UC3:** Register customer
- **UC4:** Unregister customer
- **UC5:** Associate device to customer
- **UC6:** Customize customer profile
- **UC11:** Operate actuator remotely
- **UC12:** Set alarm recipients
- **UC14:** Request consumption predictions
- **UC16:** Mark invoice paid
- **UC17:** Perform research

Most of these use cases are dependent on User Interaction. Therefor we will introduce a new component: User Interaction.

Tactics

- Performance
  - Resource arbitration
    Because consumption predictions are a computational intensive task, we will foresee a scheduler for these requests

Architectural Patterns

- Command processor
  The command processor pattern will be used for the implementation of the consumption prediction scheduler. Often the functionality of the scheduler, as described in the command processor pattern, will be split up into two separate components. These components are the buffer where the requests are stored and the actual scheduler that empties the buffer and schedules the consumption predictions. This shows a more explicit view of the workings of such a scheduler without going into another deeper level of ADD design.

Verification and refinement of drivers

None of the architectural drivers have been satisfied. Most of them are delegated to the user interaction component. Consumption predictions are delegated to the scheduler. **User Interaction**

- **UC1:** Log in
- **UC2:** Log off
- **UC3:** Register customer

- **UC4:** Unregister customer
- **UC5:** Associate device to customer
- **UC6:** Customize customer profile
- **UC11:** Operate actuator remotely
- **UC12:** Set alarm recipients
- **UC14':** Request consumption predictions
- **UC16:** Mark invoice paid
- **UC17:** Perform research

Scheduler for consumption prediction requests

- **UC14':** Request consumption predictions

Computation of consumption prediction

- **UC14':** Request consumption predictions

7.5.2 User Interaction

Architectural drivers

- **AV1':** Measurement database failure If the measurement database fails, this must be clearly indicated in the user interface (which will work through this user interaction component).
- **UC1:** Log in
- **UC2:** Log off
- **UC3:** Register customer
- **UC4:** Unregister customer
- **UC5:** Associate device to customer
- **UC6:** Customize customer profile
- **UC11:** Operate actuator remotely
- **UC12:** Set alarm recipients
- **UC14':** Request consumption predictions
- **UC16:** Mark invoice paid
- **UC17:** Perform research

Tactics

- Security
  - Authenticate users
    We will want to ensure that a user is actually who it purports to be. For this we will use a password to authenticate the users.
  - Authorize users
    Authorization of users ensures that authenticated users have the correct rights to access and/or edit certain information. These access rights may differ between different users. This access control can be by user or by user class.

Architectural Patterns

- Authorization
  We must ensure that only specific clients can access functionality of a subsystem. Therefor we will assign access rights to each client that can send service requests. to the security-sensitive subsystem and check these rights before executing any requests on the subsystem.
- Publish-Subscribe
  This pattern allows us to efficiently notify all interested parties about the modus of the system and more importantly, to be notified of changes in modus caused by other components. We will also use this pattern to notify an operator through the user interface by this publisher-subscriber pattern. The reason for notifying operators will be addressed in a further decomposition.

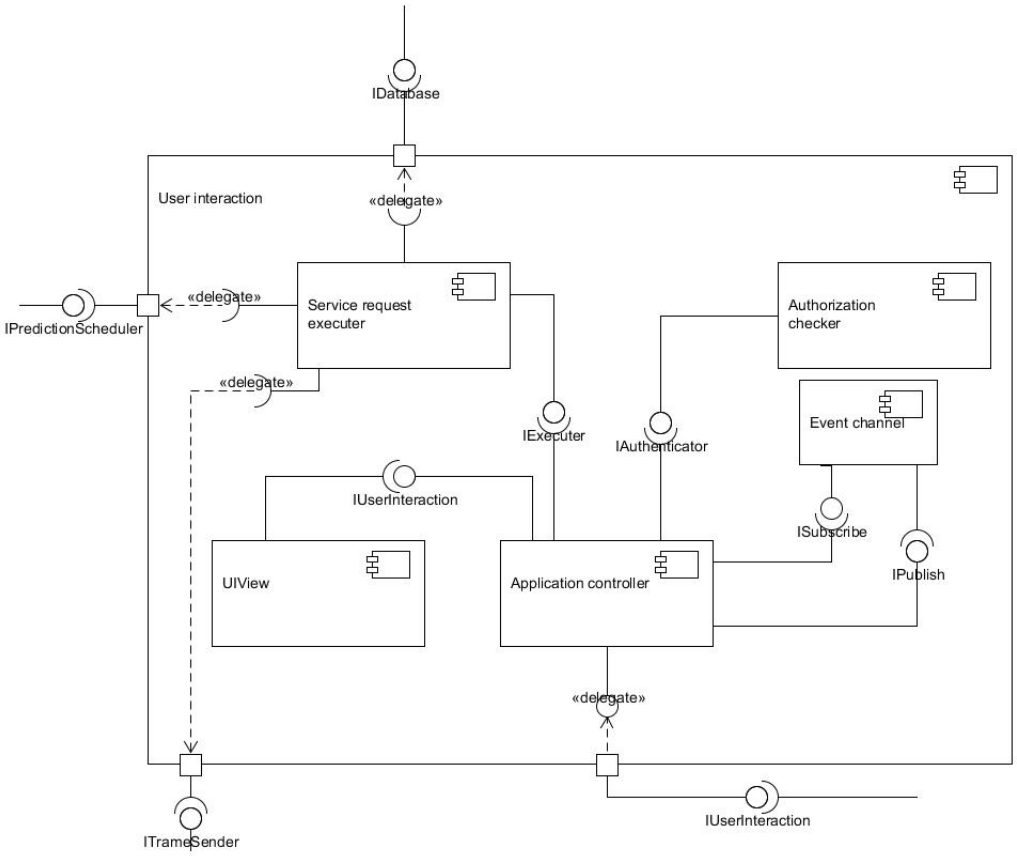

Figure 17: The User Interaction after decomposition

Verification and refinement of drivers

All drivers have been dealt with. Each of these operations is available in the user interaction interface. Before executing any of these operations, the access rights of the client requesting the service will be checked (except for UC1: Log in, here the client will be assigned certain access rights). If the measurement database is unavailable, the user interaction component will be notified of this through the event channel and will update its status appropriately.

7.5.3 Scheduler for consumption prediction requests

Architectural drivers

The main purpose of this component is making sure that new requests for consumption prediction can be saved and scheduled while other computations are still running. When a new computation can be made, the scheduler will acquire the required data from the data storage and hand this information to the computation of consumption prediction component.

- **UC14': Request consumption prediction**

Tactics

- Performance
  - Scheduling Policy
    To ensure that incoming consumption predictions are not lost and handed to the computation of consumption prediction component in FIFO order.

Architectural Patterns

- Proxy
  To increase the modifiability of the internal working of this component we chose to use the explicit interface pattern. The interface that we see outside the component will simply delegate it's methods to an internal component.
- Active Object
  The active object pattern allows us to insert new consumption prediction requests in a buffer which in its turn can be emptied by a scheduler. The scheduler can decide which requests need to be treated first. At the moment, this happens in FIFO order.

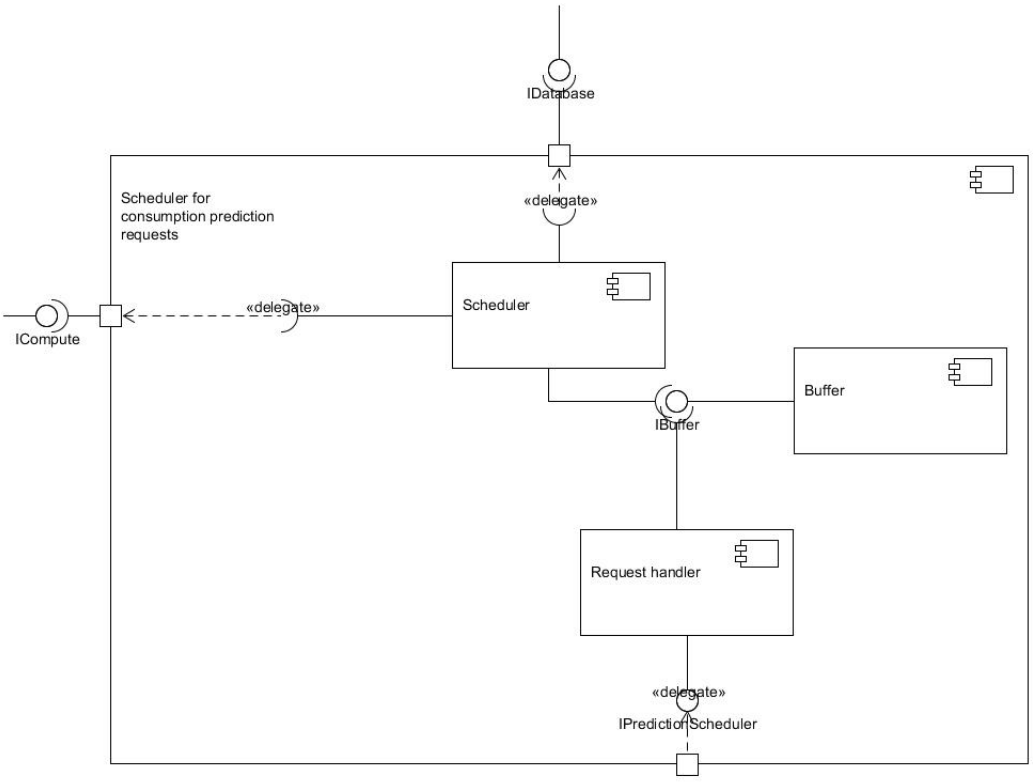

Figure 18: The Scheduler for Consumption Prediction Requests after decomposition

Verification and refinement of drivers

All the required parts of UC14' we needed to address in this component have been addressed. No further delegation is needed.

7.5.4 Computation of consumption prediction

Architectural drivers

The only purpose of this component is computing requested consumption predictions.

- **UC14': Request consumption prediction**

No further decomposition of this component is required.

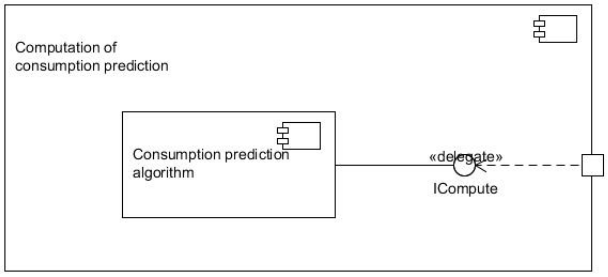

Figure 19: The Computation of Consumption Prediction component after decomposition

Verification and refinement of drivers

The processing of the data part of UC14 has been fulfilled by this component.

7.6 Assumptions

The assumption is made that a guaranteed lack of starvation is infeasible in relation to the processing of alarm trames. There are scenarios possible wherein simply cannot avoid theoretical starvation. The effects will be mitigated as well as possible but other than that, the requested properties cannot be guaranteed.

In relation to the 3 hours of measurement that need to be stored and retained when the database component should fail, the assumption is made that the buffers used, to accommodate for this measure is able to hold 3 hours worth of measurements. The size of the buffer that is needed, has to be determined further and is outside the scope of this document.

In relation to requests for the database component, the assumption is made that each request has a priority, albeit default if not otherwise explicitly stated.

Related to UC16 (Mark invoice paid), the assumption is made that the main actor indicates that the invoice has been received through the user interface. The actor will have this option in his personal portal. This action will drive the functionality for this use case.

To conform to UC15 (Generate invoice), the requirements do not indicate how and or when the generation of invoices is initiated. The assumption is made that only the client can specify the business rules for this case and it is not possible to provide an implementation without consent from the stakeholders. In the mean time, the functionality can be started from calling the generateInvoice() operation. Whichever component calls this to initiate the process, has to be specified later on. It is also possible to start this process in an event-driven manner. By monitoring the event channel, an indication that said process should be initiated, can be received.

This way of working is promoted throughout the system. It is an assumption that incoming requests are processed through a UI portal while the outgoing results are distributed directly through other channels.

In most, if not all of the significant software projects today, logging technologies are used. The assumption is made that also for ReMeS there will be a component which handles the logging information for different software components in the system. To avoid cluttering the diagrams and explanations, we omit the logger as a physical entity in our decompositions, although it actually is a useful component to have. This Logging component will not be discussed further as it is considered to be out of scope for this assignment.

When concerning the incoming gateway, the assumption is made that a physical device for receiving trames from different networks is installed already and that this device delivers the incoming trames to the incoming gateway. Normally the same boundaries should be assumed as in the outgoing gateway, but due to unforeseen circumstances, this wasn't addressed in the decomposition of the incoming gateway component.

In the case that emergency services need to be notified because of a gas leak, it was not quite clear how this notification should occur. The assumption is made that an sms will be sent to the emergency systems to notify them of the problem.

7.7 Remarks

When using the ADD process to create an architecture for the ReMeS system, the dept of the iterations was limited to two levels in most cases. In some cases, however, another decomposition (level 3) was needed to explain some concepts a bit better.

The concept of iterations was used to describe different phases of the ADD project. An iteration is seen as a complete decomposition to the lowest level starting from the highest level. An iteration ends when there are no more decompositions to be done on a lower level while still dealing with the original top level components. Each time the component called Other functionality gets decomposed to address some previously unresolved quality attributes and/or use cases, a new iteration begins.

It is however the case in this assignment that a limited amount of quality attributes have been supplied to use as main drivers for the add process. Due to this, some functional requirements can not be linked or deduced from the supplied quality attributes. When this is the case, those functional requirements will not be used to instantiate components in the architecture by using the ADD process alone, since ADD stands for Attribute Driven Design. Most of these use cases are related to the user interface. There are no given quality attributes addressing any of the user interface functionalities. These functionalities will be added (albeit in a lower level of detail) in the last iteration and in the final architecture described in section 8. It is however not a strict ADD-methodology that is used for those remaining functions.

In some of the lowest level decompositions, the decomposition seems to be driven solely by functional requirements without quality attributes. Sometimes it is possible that the non functional drivers for an iteration are already met but another decomposition for child components is needed to illustrate some remaining functional requirements. This is situation is however different from what is discussed in the previous paragraph where there were no non-functional drivers to begin with.

8 Final Architecture Design

8.1 Context diagram

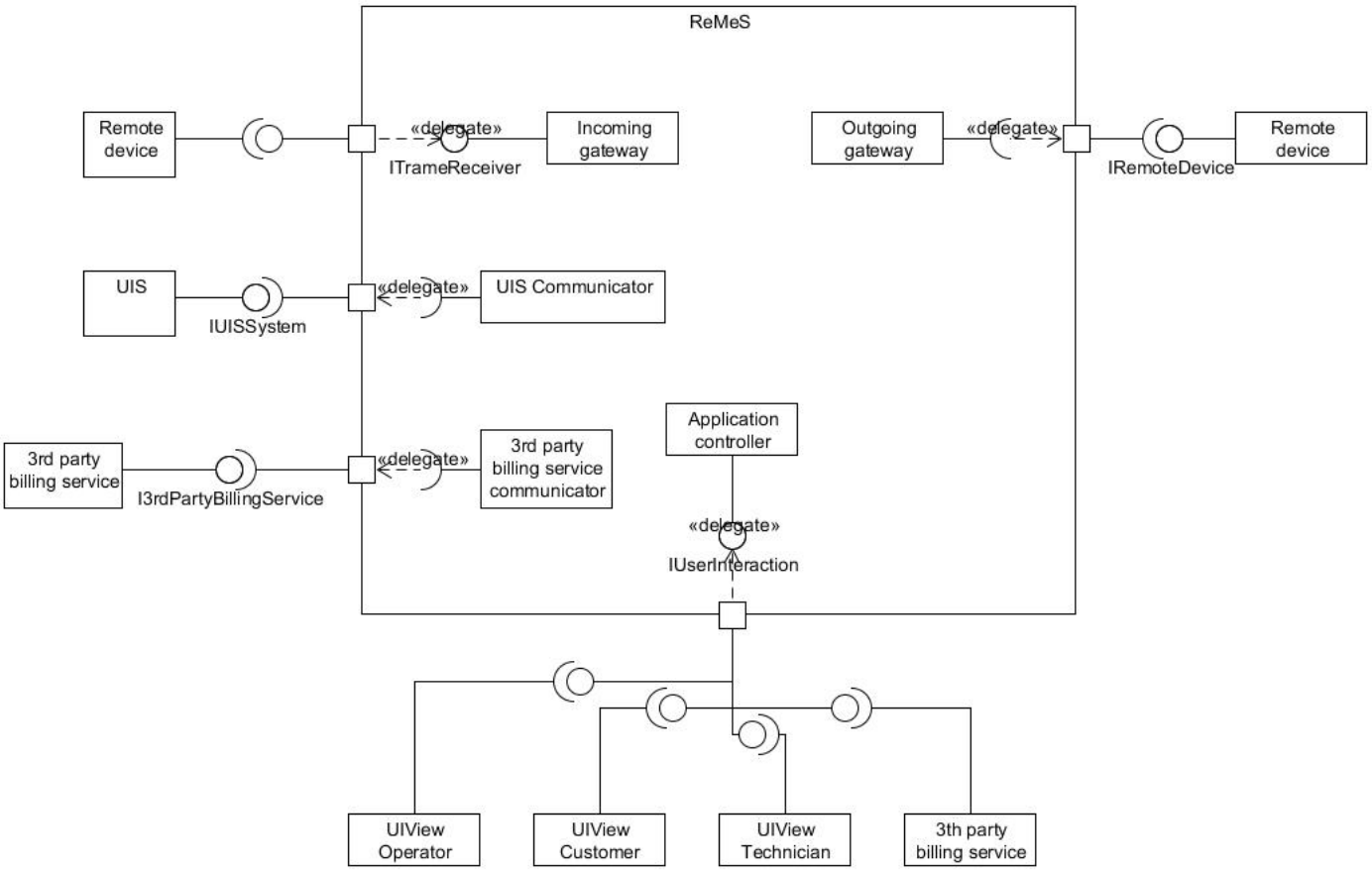

Figure 20: The context diagram for ReMeS.

The context diagram is used to describe the most important external factors that interact with the system. In this case the most important system to view is the ReMeS system itself. This is the system that has been designing thus far.

The most important external actors are the remote device, the users and the UIS (User Information System). The diagram is in fact quite self-explanatory as to how these components use the border components in the ReMeS system in question. The interfaces through which these forms of communication happen, are also included in the context diagram, albeit in name only. The detailed version of these interfaces can be found in section 12.2

8.2 Overall component diagram

The overall component diagram provides a good depiction of how the ReMeS is divided into components that together handle the functional and non-functional requirements.

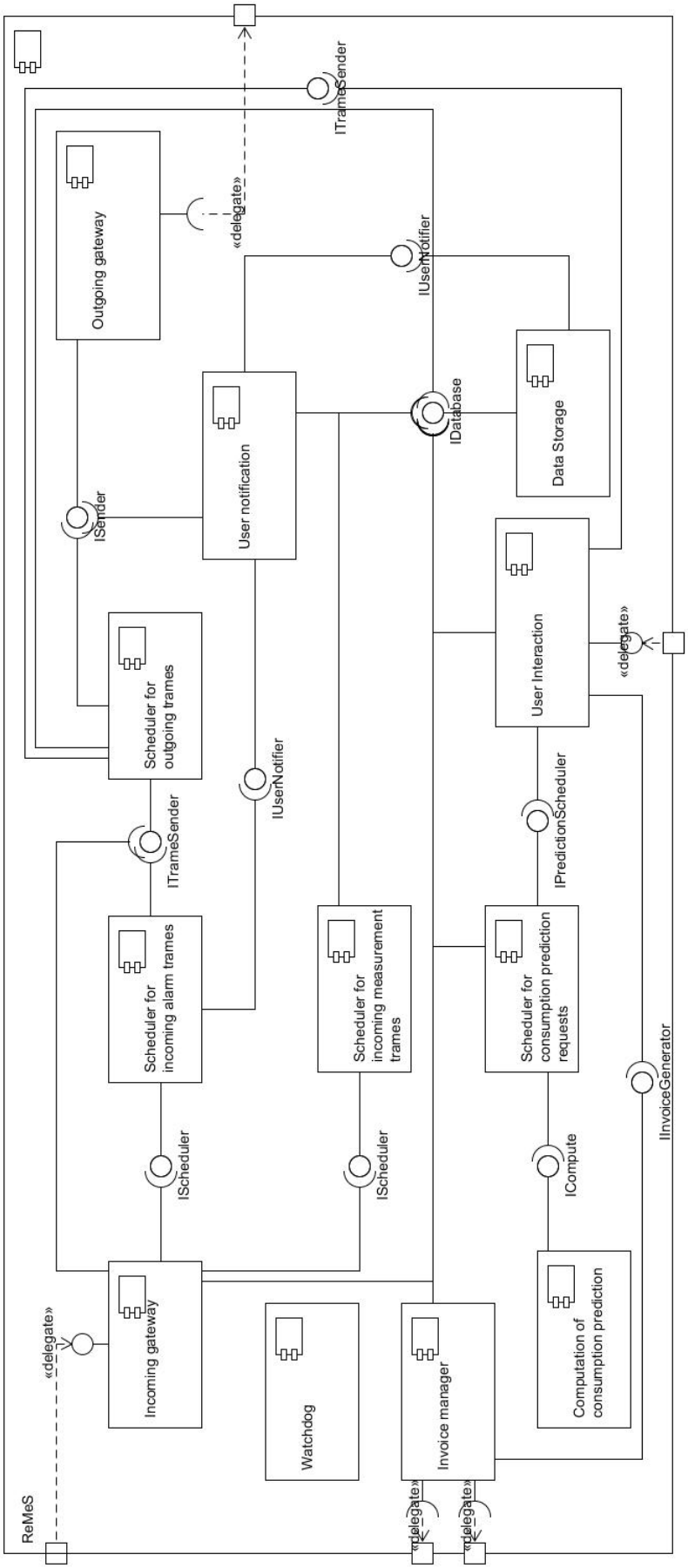

Figure 21: The final architectural diagram for ReMeS.

This section will explain how the core functionality is supported by the provided architecture. The diagram shown in figure 21 is a depiction of the architecture that came forth out of the previously discussed ADD runs (with a few minor modifications afterwards).

There are a couple of major components that play a vital role in this architecture. These components will be explained further in the following section about the decompositions. One of the most important components shown in the diagram is the data storage component. Most of the components use this data storage component in one way or another. That is why the interface the data storage component provides, is a requirement for a lot of different components. This also shows in the diagram if one follows the lines going out from the IDatabase.

The Watchdog component that is present in the overall component diagram doesn't seem to interact with the rest of the components. The watchdog component however has interaction and communication functionality with every other component. All components for which failure needs to be detected, will send heartbeat updates to this central watchdog. We omitted drawing these interface dependencies because of the fact that the whole diagram would have become practically unreadable. Keep in mind though that this is not some ghost component with no interaction with any component.

It is also important to note that an event channel is used throughout the system. This event channel is used for the dispatching of different types of messages such as failure indications and component state updates. This component is actually not shown in the overall component diagram because it interacts with a lot of different components but on a lower level than the overall component diagram is meant to show. This event channel will be shown and explained in both the further decompositions in section 8.3 and the element catalog in section 12.1

8.2.1 Core functionality

The core functionality of the system can be described as the ability to handle measurement information utility networks in the form of trames sent by remote devices. Also the control of remote devices installed on valves, is an important core functionality. Also the management of and reporting of this data by and to the user, is an important aspect of the functionality this ReMeS system should offer.

The proposed architecture uses an incoming and an outgoing gateway to handle functionality of receiving and sending the data trames from and to the remote devices. The incoming gateway is able to route different types of trames to different schedulers. These schedulers decide when, where and how these trames should be processed. Eventually the information is extracted from the trames and stored in a data storage component.

The different schedulers and processors also have the ability to schedule the notification of users (under certain circumstances such as leaks or failures) through a user notification module. The next section will handle the different components in a little bit more detail.

After the processing and storage of an incoming trame it may be the case that control trames should be sent automatically to effect certain effect in the remote devices. The components responsible for the processing of the incoming trames have the option of scheduling these outgoing trames when necessary. These outgoing trames will be sent through the outgoing gateway.

The former describes the main data flow for trames throughout the system. Another important aspect is the user interaction. A user interaction component handles all user input and commands to the system. This component provides views of the ReMeS state based on the type of user interacting with the component and handles the interaction of this user with the system. When a user issues a command, it can interact with the data storage component, but also the effectuation of sending control trames will be handled by the system through a manner of dispatching the commands to the right component.

A last important aspect is the use of an invoice manager which is in charge for the generation of invoices based on triggers by either the UI (for an operator) or certain internal events.

8.3 Decomposition component diagrams

This section will handle the decompositions of the major components in the overall component diagram. However a lot of decompositions are illustrated, discussed and explained in the ADD sections. To avoid overly repeating ourselves, the modules that have not changed since the last ADD iteration, will only be referred to.

The incoming gateway is shown in figure 4. This is the gateway which receives the data trames from remote devices. This component also handles the acknowledgements of incoming trames and the callback for expected acknowledgement from sent out trames.

The scheduler for incoming measurement trames is shown in figure 5. This component handles the incoming measurement trames and schedules them for processing. This component also handles the checking for missing measurements in the trame handler. The buffer in this components handles the storage of measurement data in case of a central database failure. This buffer is made redundant by using multiple buffering instances separated on a network. The scheduler component can use different modes of scheduling based on the operation mode and workload that is being advertised on the event channel.

The data storage component is shown in figure 7. This component handles the central data storage and effective processing of measurements. The DB request handler handles the requests for access to the database and also incorporates a caching mechanism for availability and performance. This component internally uses different physical databases for different types and natures of data. Anomaly detection is also provided by this module. These physical database are internally shielded by a database access layer for object oriented interaction with the databases.

The watchdog component is shown in figure 8. This component provides the availability monitoring service for the system. This module uses the event channel to post messages about the uptime status and possible failures to all interested (or subscribed) parties. Each component that wants or needs its status monitored sends repeated measurements to the watchdog to notify this component of it's activity (in some circles described as petting the watchdog). A missing heartbeat will be flagged by the active loop and in that manner the failure of a component can be detected.

The user interaction component is shown in figure 17. The user interaction component is the component that provides user interaction for this system. It is realised by using a MVC (model-view-controller) structure. Each user category has its own type of view on the system. These views will interact with the controller layer to perform actions and get information from the core system. The application can request data by directly querying the database component or by monitoring the event channel of the system.

The outgoing gateway is the only component that has really changed since our last iteration of ADD. During the ADD process, the different manners of sending data was made abstraction of. Eventually, for the final architecture discussion, it was decided to elaborate on the different subcomponents that are used to send trames over different technology networks (such as sms, grps or ethernet). The final structure of the outgoing gateway is shown in the following figure.

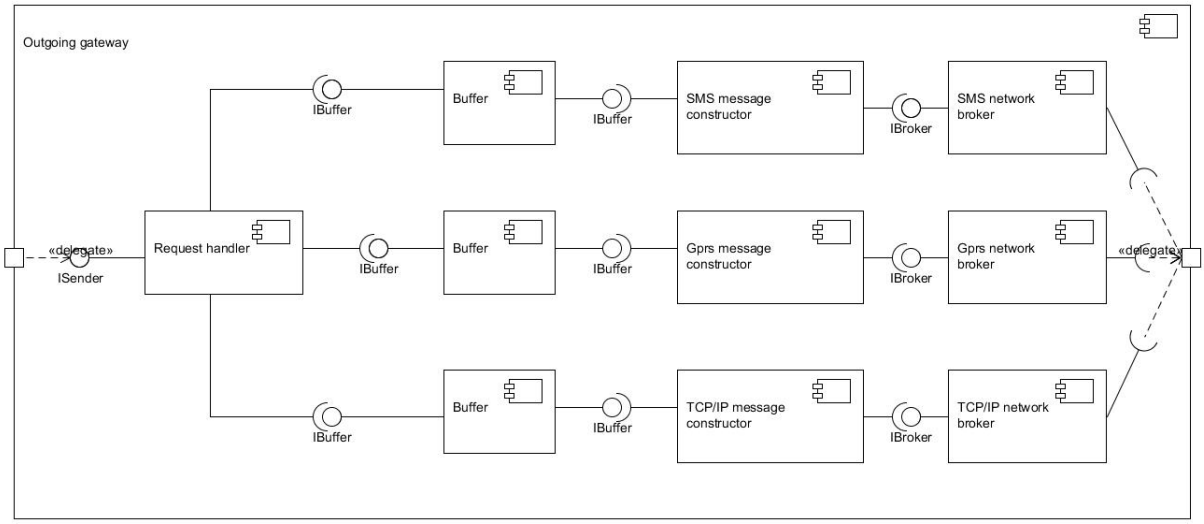

Figure 22: The component diagram of the final version of the outgoing gateway component.

The rest of the components are of a more trivial nature and the explanation and discussion can be found directly in the ADD section.

8.4 Deployment diagram

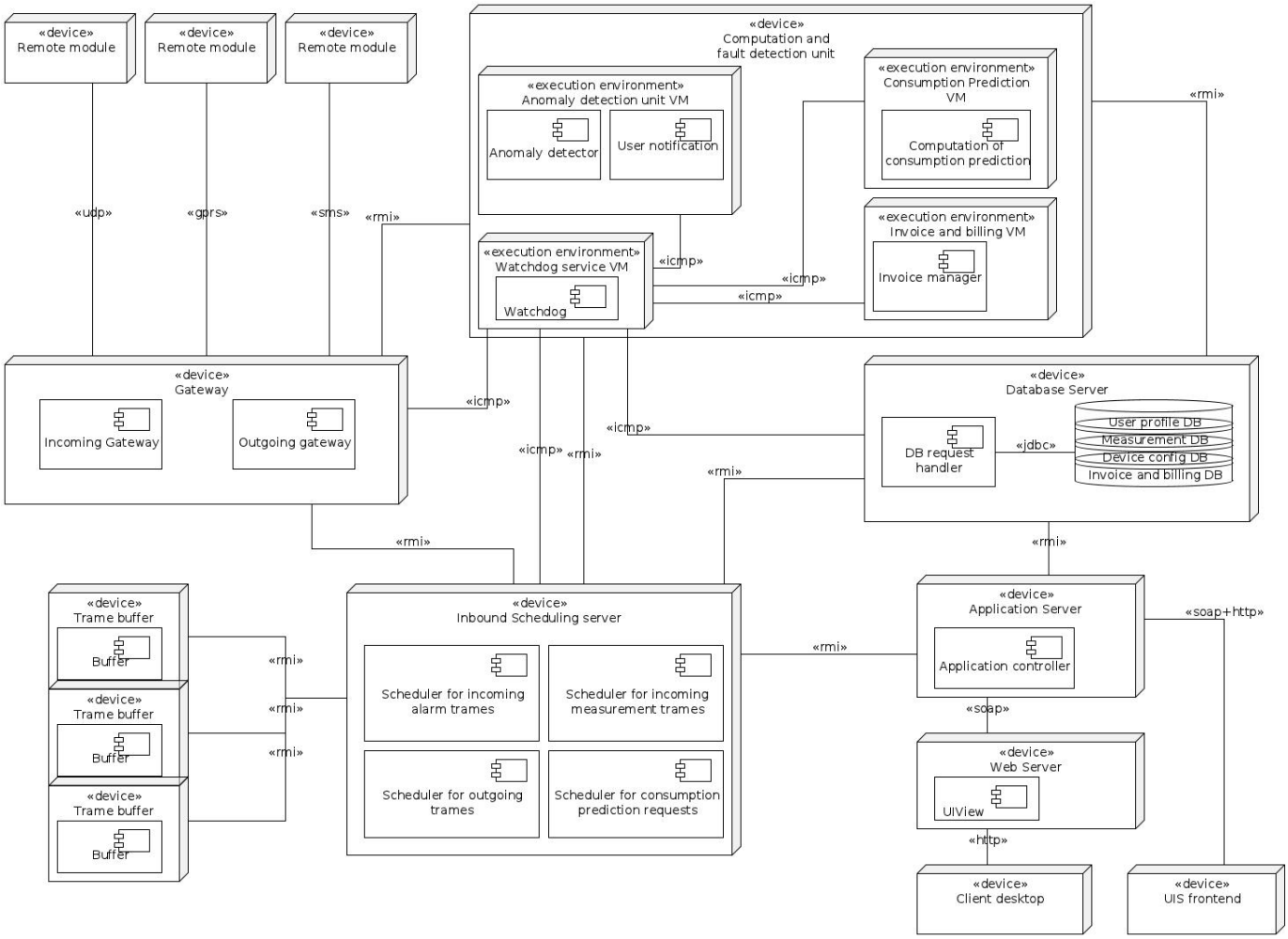

Figure 23: The final version of the ReMeS system deployment diagram.

The deployment diagram shows how different architectural components are to be deployed in a real world scenario. There are 4 major devices that make up the ReMeS system, not counting the application server and web server while two major communication protocols are used internally.

8.4.1 Device Nodes

These devices are the Gateway which forms the physical link to the outside world when speaking about trames. This device handles all communication to and from the remote modules, be it over an IP channel, a gprs channel or an sms channel.

The next component in line is the inbound scheduling server. This device handles the scheduling of trames coming in from the gateway or the trames pending to be sent out to remote devices. Also the scheduling for processing the measurement trames and the scheduling of consumption prediction requests gets handled by this device.

This device also has a connection to three standalone buffering devices. These buffers are very important for redundancy and availability purposes.

The next component is the consumption prediction server. This device handles the generation of consumption predictions based on the user history and measurement data.

The next component in line is the Database server. This server is an important device in the ReMeS system because this device will hold just about all the data albeit stored in different databases. Request to the database are handled through the request handler component which communicates with the database instances themselves using the jdbc protocol. This device is also the device with the most coupling to other components in the system. This is quite normal because the whole system is built for the core functionality of storing, transforming and using this data in one way or another.

The next component is the Computation and fault detection unit. This device is actually a device which hosts and supports multiple execution environments. In reality this could be realised by hosting multiple virtual machines layered on top of the host operation system running on this device. The first execution environment has as responsibility the detection of anomalies in the incoming user data and the notification of users when such anomalies are detected. The second execution environment provides the watchdog service that is being used to monitor the uptime of different components in the ReMeS system. Heartbeats are sent to the watchdog service by different components to inform it that they are still up and running. The heartbeats are sent using the icmp protocol. The third execution environment provides the computation of consumption predictions. The fact that consumption prediction component resides inside a virtual execution environment allows the replication of this component for load balancing purposes. If more consumption prediction modules are needed, it just suffices to replicate the virtual machine on different physical computation nodes that are capable of running virtual machines. This improves the scalability and load distribution aspects for the computation of consumption prediction. The fourth and final execution environment hosts the invoice manager. This component is capable of generating and dispatching invoices for customers and clients alike.

Beside these major devices, also an application server and a web server will be deployed in order to provide the human users and the User Information System with an easy to use interface to use the functionality provided by the ReMeS system. The web server will use the SOAP protocol to communicate with the application server and outside users can visit the web pages provided by this web server by using the http protocol.

8.4.2 Communication protocols

For the main method of communication between the different devices, the rmi protocol will be used. This form of communication will be made possible by connecting the different devices to a switched network based on the IP protocol. On top of this switched network, the rmi protocol will provide a means of communication for the internal components of the devices. Besides rmi, the icmp protocol will also be used by a lot of devices, to conform to the watchdog service specifications. The icmp protocol will be used to send heartbeats to the watchdog service for uptime monitoring purposes.

Furthermore two protocols for accessing the service from outside the ReMeS system will be used. The soap and the http protocol are both able to be transfered on the previously discussed switched network.

Finally, the remote devices have different options for communicating with the ReMeS system. They can use a simple udp, but also die gprs and sms protocol can be used to send and receive trames to and from the ReMeS system.

9 Scenarios

This section will handle the behavioral discussion for the ReMeS system. This section will illustrate, using a set of preset scenarios, how the system and its architecture support the fulfillment of these scenarios.

9.1 Notes

In a number of diagrams the auth() function call is used to authorise a user for an operation. However, in the diagrams the auth() function call is passed only an authentication token object. In reality this function call would need an authentication token and an object representing the operation that an actor wishes to execute in order to successfully authenticate that actor.

Another point of attention are the representation of the function calls made to the data storage instance. These calls are described and shown on a very high level of abstraction. Representing these interaction in a more detailed manner would very quickly lead to unnecessarily complex diagrams. Therefor this more simple approach is used. In reality also a query would be passed as an argument to execute operations on a database in stead of just the information that is needed for the operation (which is done in the sequence diagrams here).

9.2 User profile creation

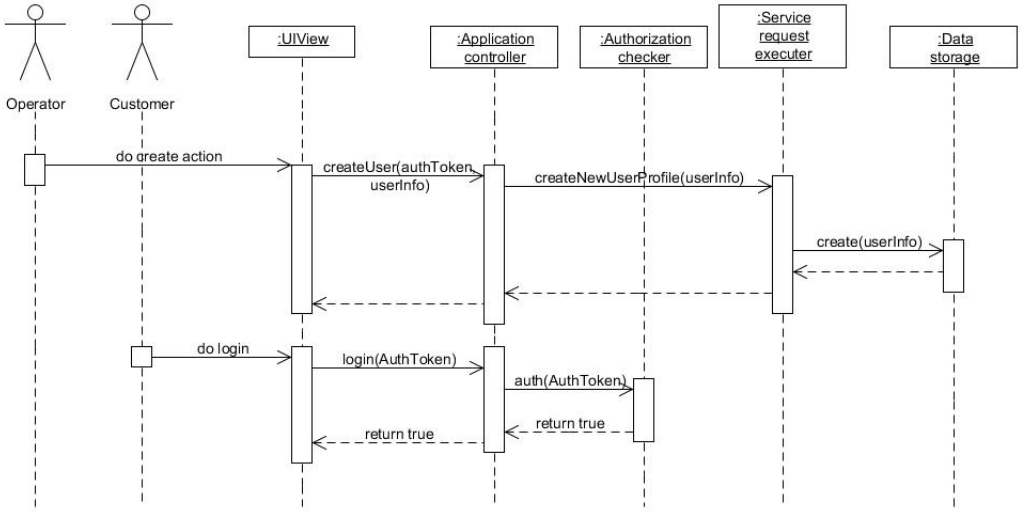

Figure 24: The sequence diagram for the user profile creation scenario.

The user profile creation scenario is mainly handled by the user interaction component. In the sequence diagram only the UIview component is shown. In a real scenario, this component would be divided in separate views for normal users and operators. Further decomposition of this module was not warranted in the ADD phase. Even though it is not shown in the sequence diagram, the operator must first login into the application controller. It is also not shown that when the operator calls the createUser() method, the application controller will first check if the current user is authorized to execute this action.

9.3 User profile association with remote monitoring module

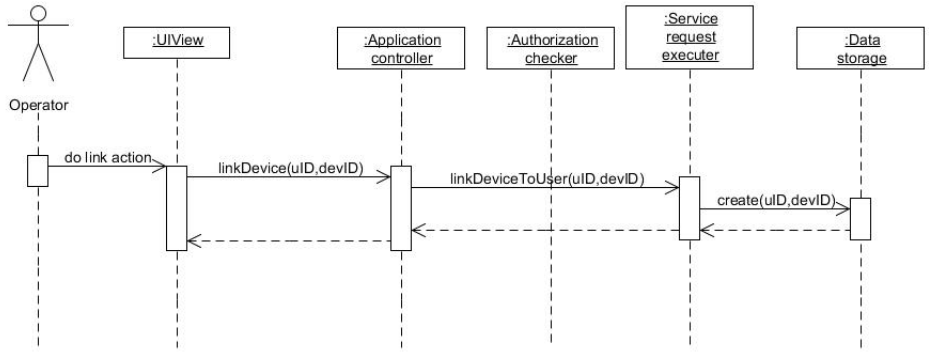

Figure 25: The sequence diagram for the association of the device to the profile.

This scenario illustrates that the functionality for association of a device with a user profile is provided. However the notification of the UIS is not foreseen. At least not in an elegant manner. Due to the late addition of the components in charge for communication with the UIS, this functional requirement was overseen. Again the UIView an actor can use will be tailored to the actual type of actor.

9.4 Installation and initialization

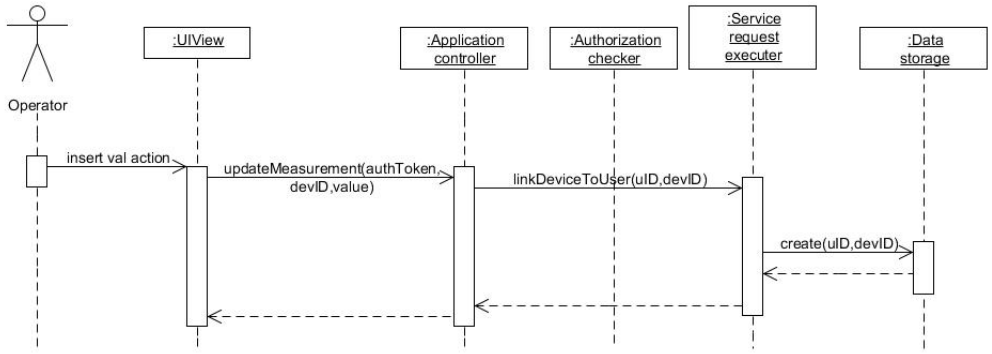

Figure 26: The sequence diagram for manually setting the meter value for a device.

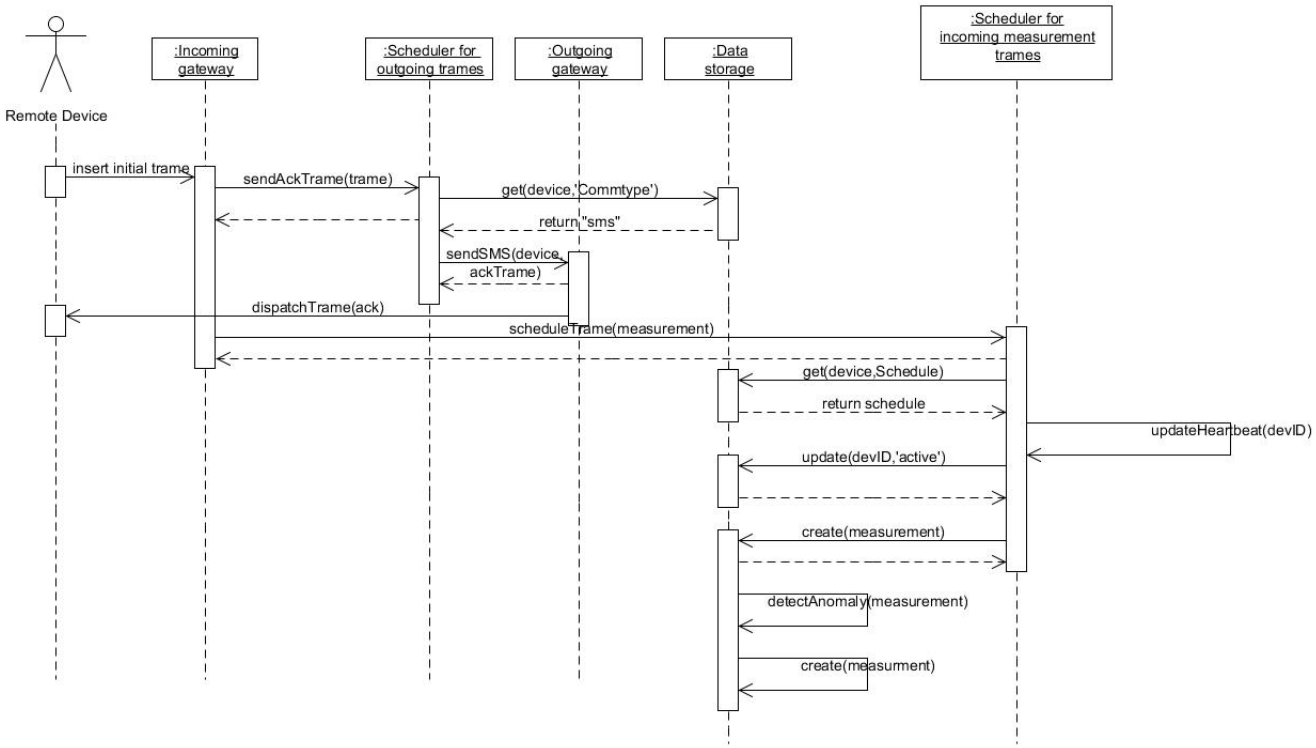

Figure 27: The sequence diagram for sending the initial measurement to ReMeS.

The scenario of installing a remote module at a remote location can be split up into two parts in order to discuss this behavior. The first functionality lies with the operator actually manually setting the initial meter setting through the user interface he presumably access through the web browser of his mobile unit. Again, in this sequence diagram it is not shown that the operator has to login into the application controller and that the checkRights() method is called by the Application controller.

The second behavior, which is described in the second sequence diagram, shows how the initial measurement is sent to the ReMeS system and how it is passed along the system until it is stored and until the device is marked as active. The sequence diagram also show the function calls to some components returning immediately. This is done on purpose to indicate that messages are being passed (through the use of a buffering mechanism) to other components in a way such that the sender doesn't have to wait for other operations to complete. Each component has it's own responsibilities and thus, only adhere to their own operations.

9.5 Transmission frequency reconfiguration

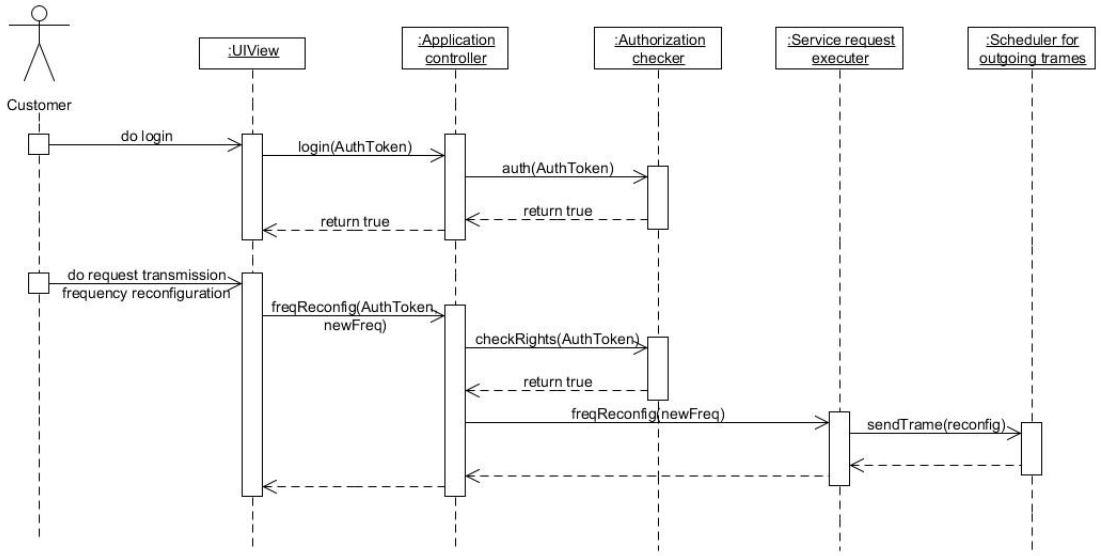

Figure 28: The sequence diagram for reconfiguring the transmission frequency.

The transmission frequency reconfiguration scenario is almost completely handled by the User interaction component of the system. This system diagram shows that the complete functionality of the scenario is provided by our system.

9.6 Troubleshooting

The scenario described in this section mainly describes interaction between actors in the problem domain. For the ReMeS system under development, this scenario does not bring much new functionality to describe. Therefor we didn't explicitly work out this example since the result would be too similar too other scenarios.

9.7 Alarm notification recipient configuration

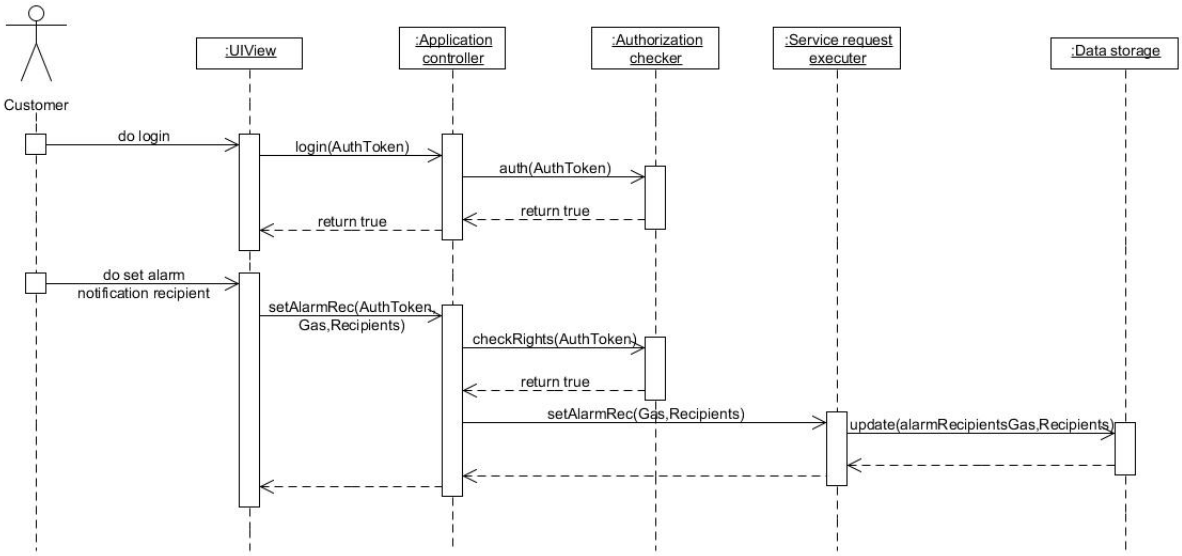

Figure 29: The sequence diagram for configuring the alarm notification recipient.

This scenario is very similar to the transmission frequency reconfiguration scenario. It is also almost completely handled by the User interaction component of the system. The only difference is that the service request executer will communicate with the Data storage component instead of the Scheduler for outgoing trames. The full functionality of the scenario is provided by the system.

9.8 Remote control

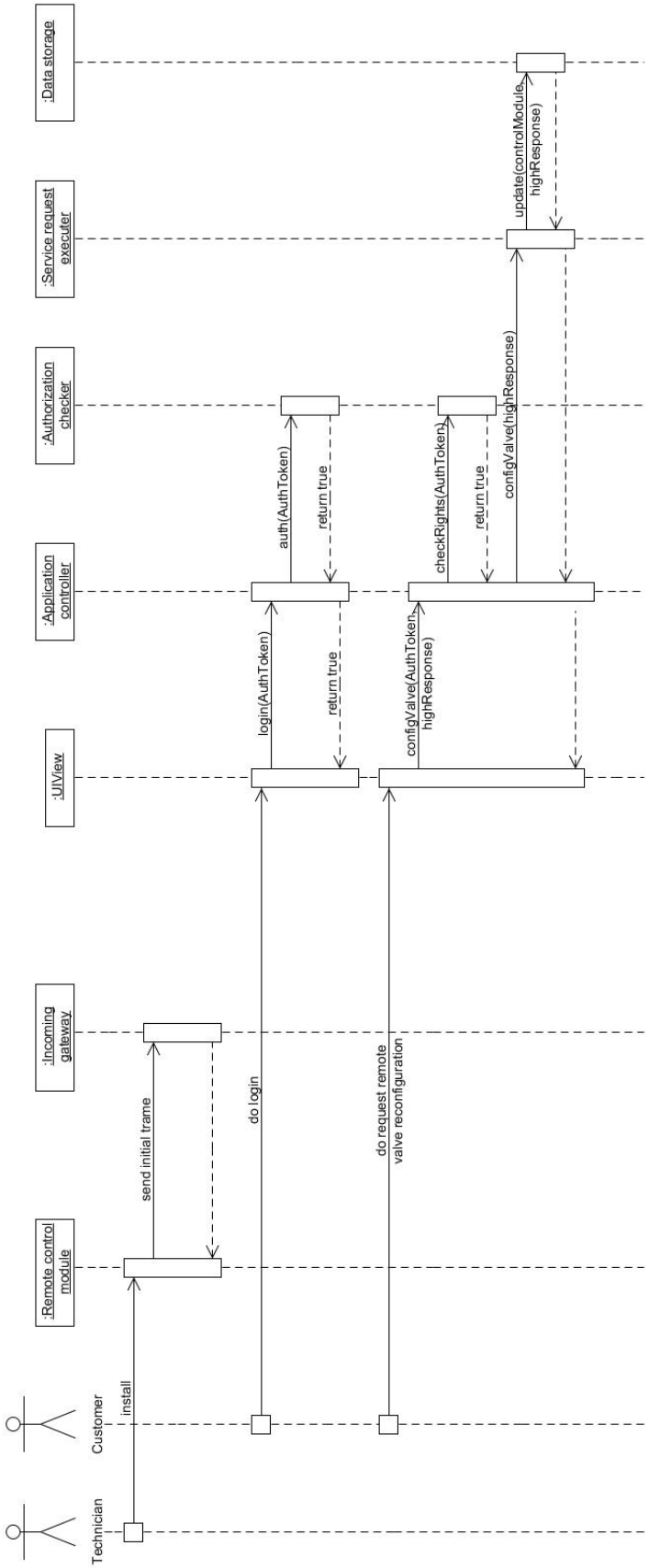

Figure 30: The sequence diagram for remotely configuring the remote device..

The first part of the scenario Remote control is the installation and activation of a remote control module. Unfortunately there is no functionality in the current ReMeS system to activate a remote control module. Only remote monitoring modules are activated automatically. It is possible to activate a remote control module through the User interaction component. That way, an operator can do this. The second part of the scenario is configuring the water control module. This part is very similar to the scenario Alarm notification recipient configuration. A customer must use the Application controller to select the desired configuration profile.

9.9 Normal measurement data transmission

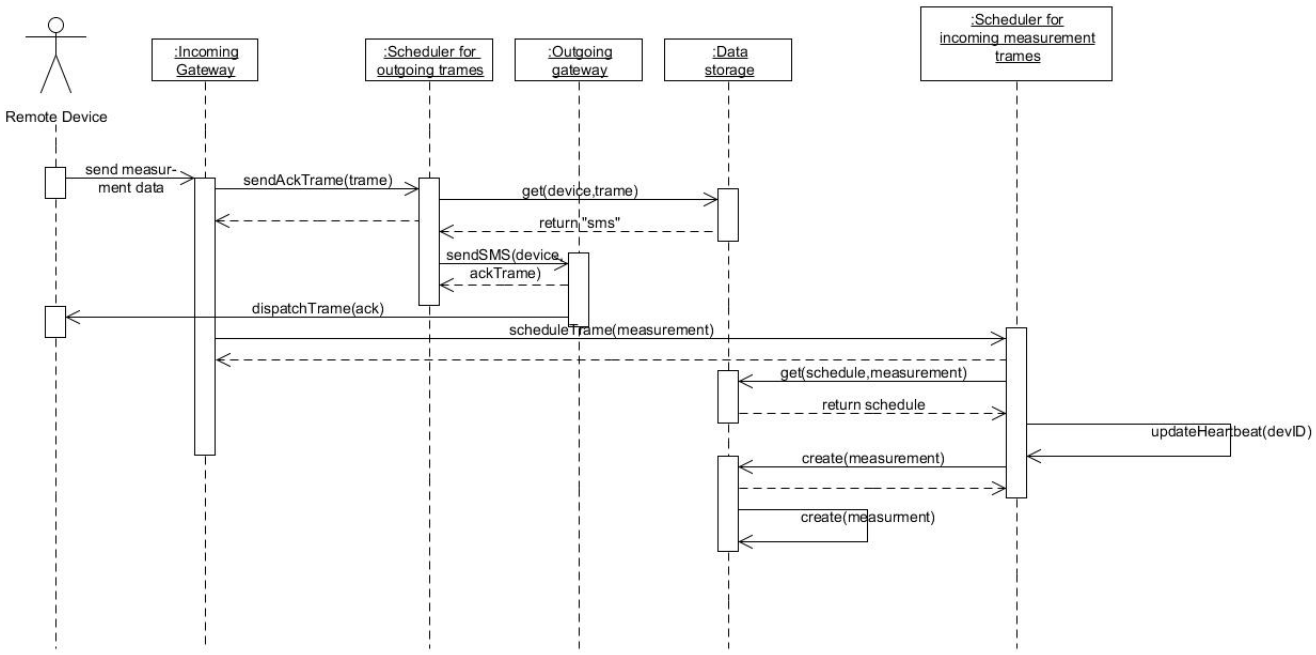

Figure 31: The sequence diagram for sending a measurement trame under normal circumstances.

This sequence diagram shows that the Incoming gateway will send an acknowledgement through the Scheduler for outgoing trames to the remote device. The Incoming gateway will forward the measurement trame to the Scheduler for incoming measurement trames. This component will do all required tasks to store the measurement in the Data storage. The Data storage component will also call the Anomaly detection component. This is not shown in the sequence diagram because it isn't in this scenario.

9.10 Individual data analysis

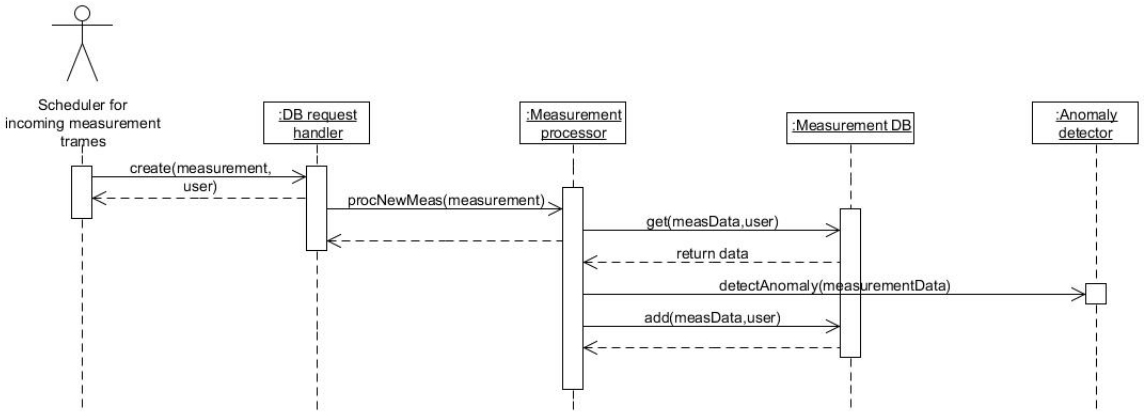

Figure 32: The sequence diagram for performing individual data analysis requests.

When the Data storage component receives a new measurement to store, it will call the anomaly detector to compute individual consumption profiles. These profiles can be used to determine potential leaks or anomalies. The complete scenario is covered by the design of ReMeS.

9.11 Utility production planning analysis

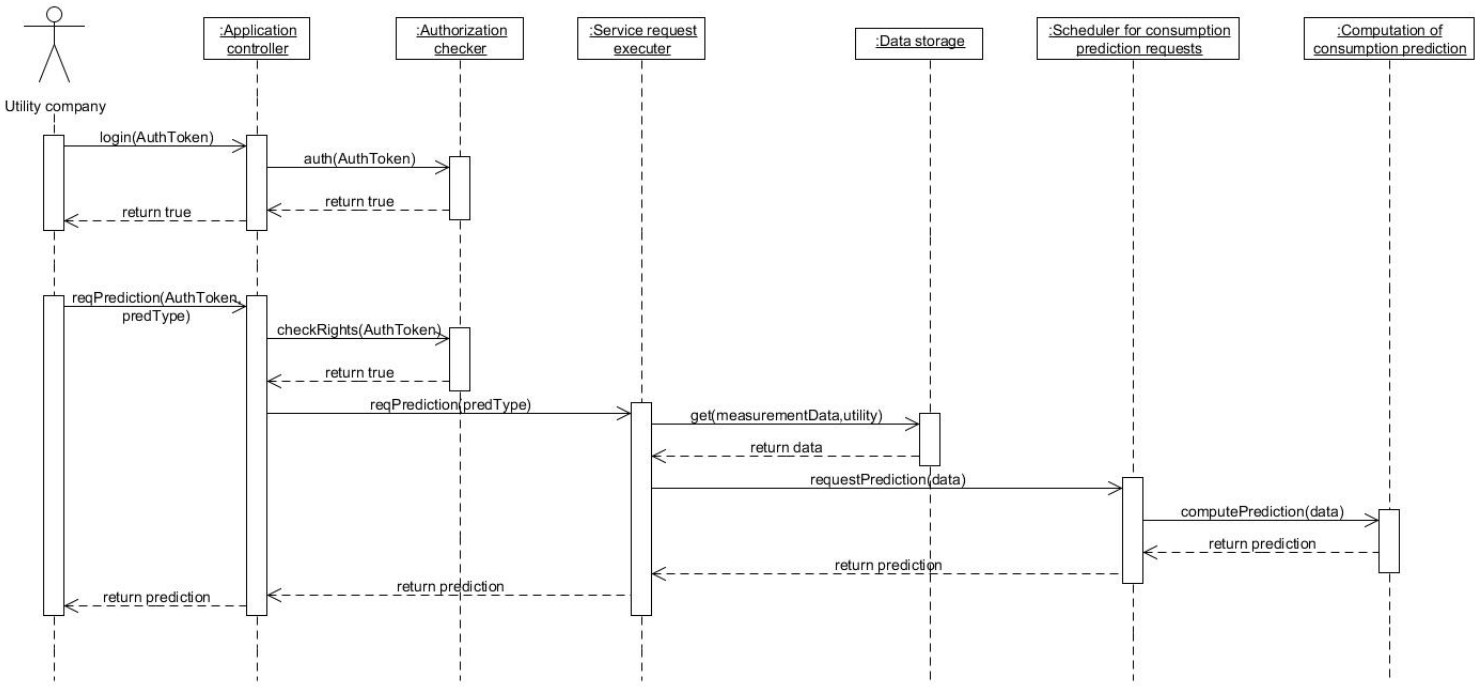

Figure 33: The sequence diagram for performing production planning analysis.

Utility production plannings can be requested through the Application control Component (or through the UIView component). Since utility companies will probably want to automate these

requests, we provided an interface they can use without going through the UIView component. If a utility production planning is requested, this request will be added to the Scheduler for consumption prediction requests. When the computation of the prediction is finished, it will be returned to the caller. This scenario is also completely covered by our design.

9.12 Information exchange towards the UIS

This scenario indicates the regular information exchange between UIS and ReMeS. This functionality is available in the architectural design but only at such an high level so that creating sequence diagrams to illustrate this, would result in a fairly trivial diagram. These modules were not decomposed very deeply and that is why it is not possible at the moment to show a much more detailed view of this behavioral aspect of the design.

9.13 Alarm data transmission: remote monitoring module

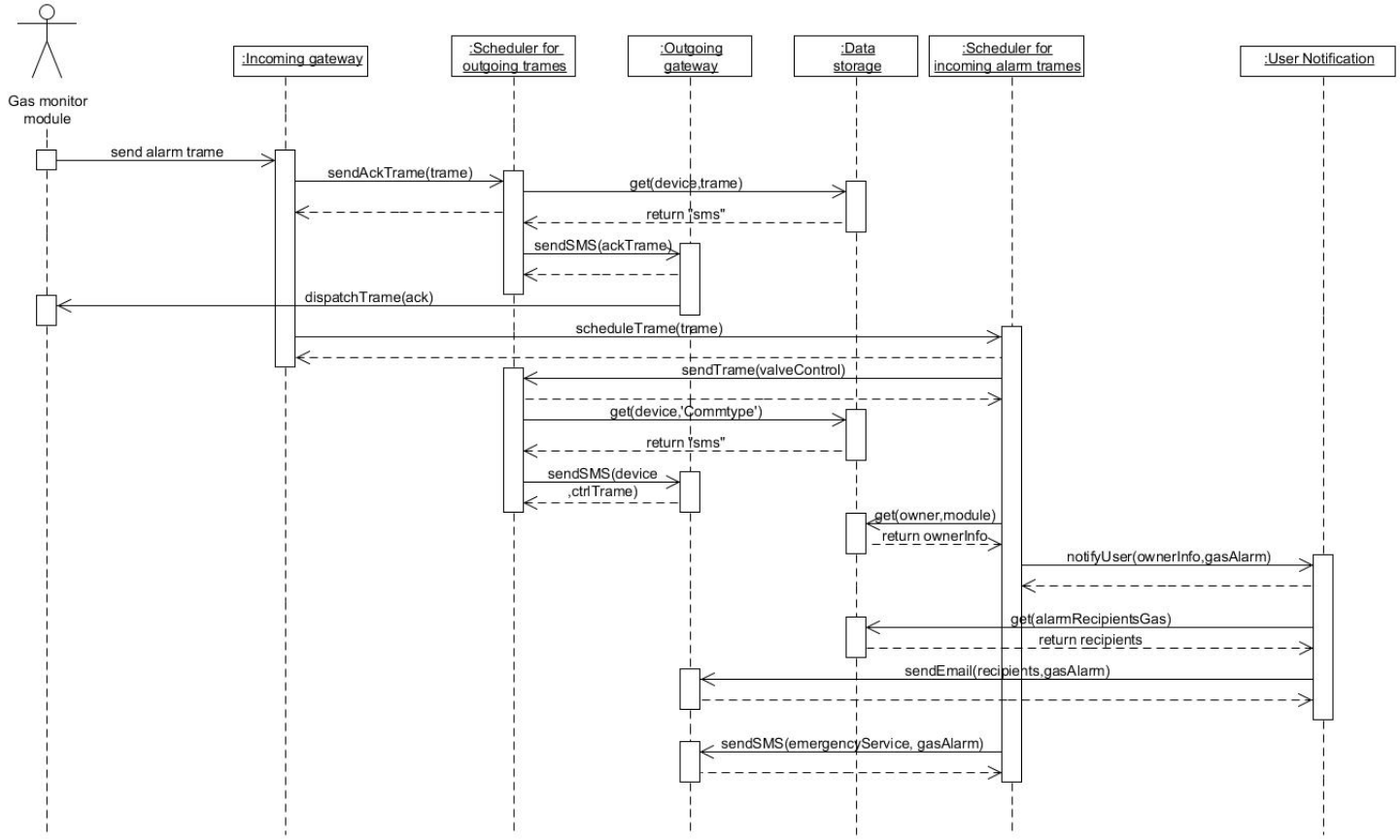

Figure 34: The sequence diagram for receiving an alarm trame.

When an alarm trame arrives at the Incoming gateway component. This trame will be delegated to the Scheduler for incoming alarm trames. This component will send the required notifications to the correct modules, people and emergency services. Since we didn't know how we should contact the emergency services through an automated method, we used the SMS service to notify the emergency services. Keep in mind that in the sequence diagram, all the delegations made by the Scheduler for incoming alarm trames are executed concurrently.

9.14 Alarm data transmission: ReMeS

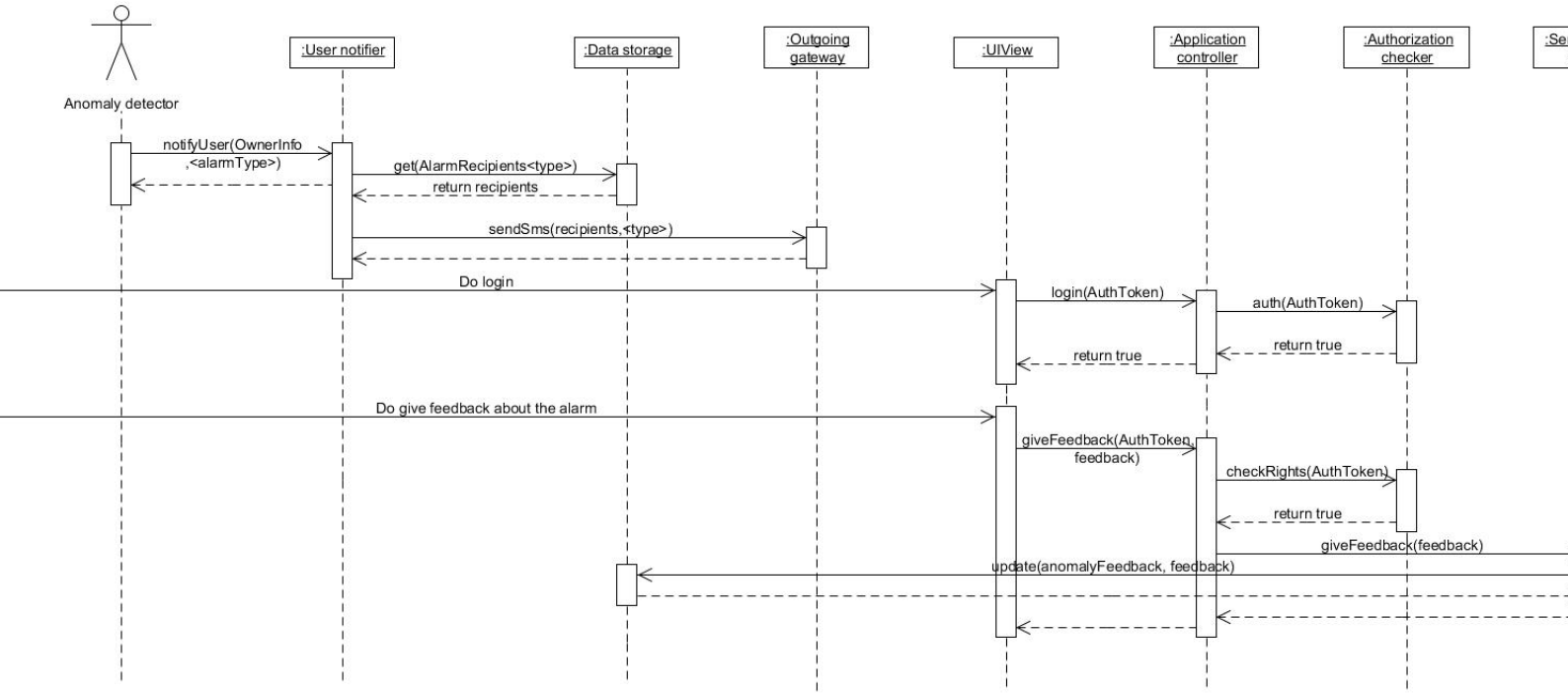

Figure 35: The sequence diagram for detecting an anomaly.

This scenario is also completely covered by the ReMeS system. When the anomaly detector detects an anomaly, it will call the User notifier component to send a message to the appropriate customer. The customer can also login into the UIView to give feedback about the detected anomaly. The customer can give positive feedback (the anomaly was indeed a leak) or negative feedback (the anomaly was a false positive).

9.15 Remote control module de-activation

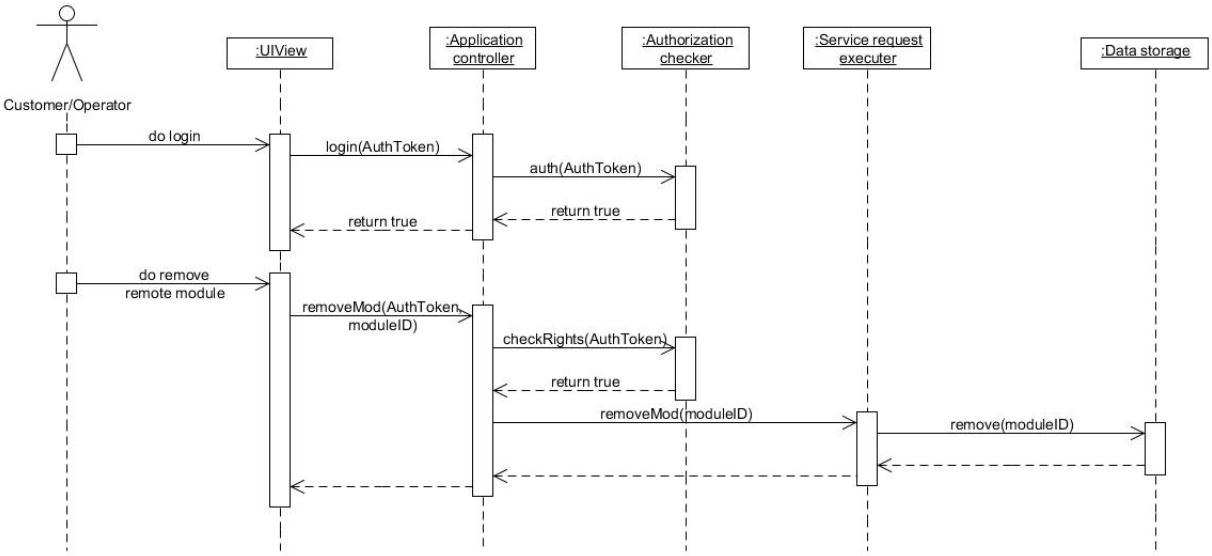

Figure 36: The sequence diagram for removing a remote module from a customer account.

This scenario is again very similar to the Alarm notification recipient configuration scenario. This action is available through the Application controller. One needs to login first before removing the remote module from the Data storage.

9.16 New bill creation

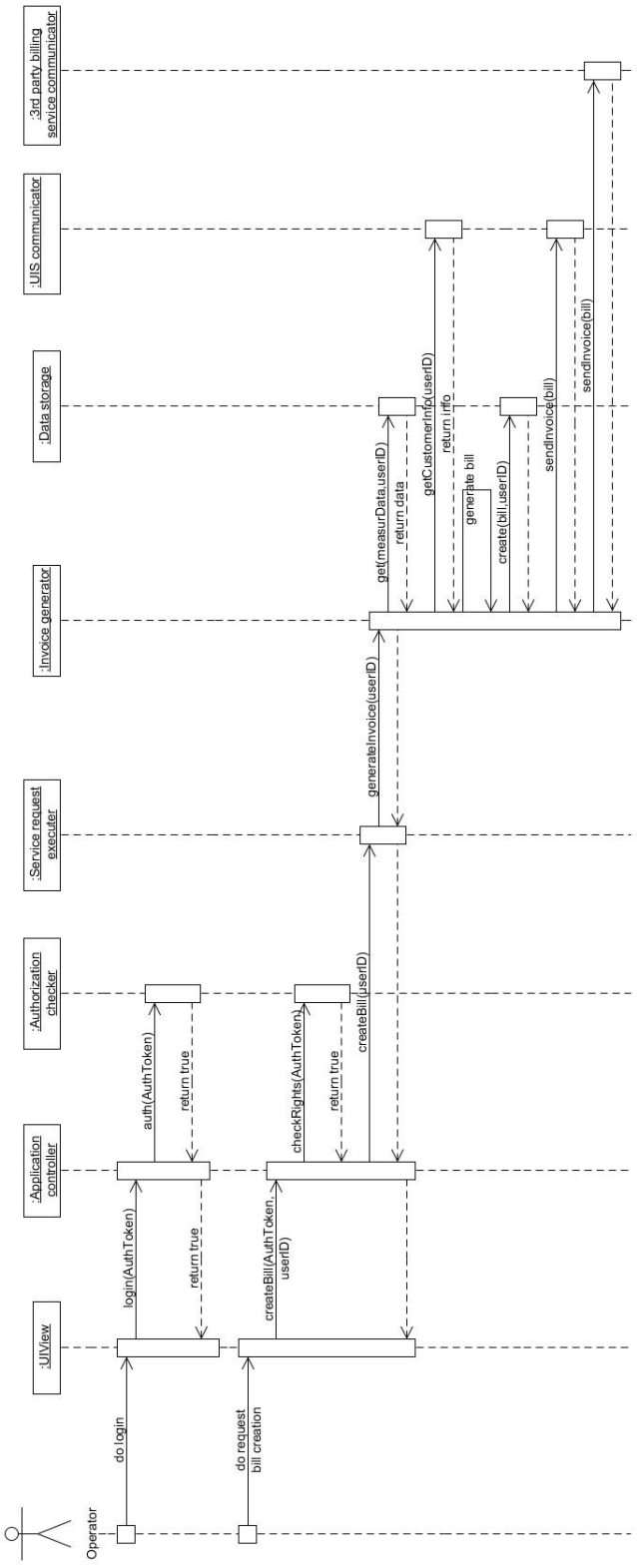

Figure 37: The sequence diagram for creating a new invoice.

Creating a bill is fully implemented in our ReMeS system. The only part that is different from the Bill Creation scenario is that ReMeS does not automatically creates a bill, but that an operator must initiate the operation. The created bill will be sent to a third party billing service and to the utility providing company. The created bill will also be stored in the Data storage component.

9.17 Bill payment is received

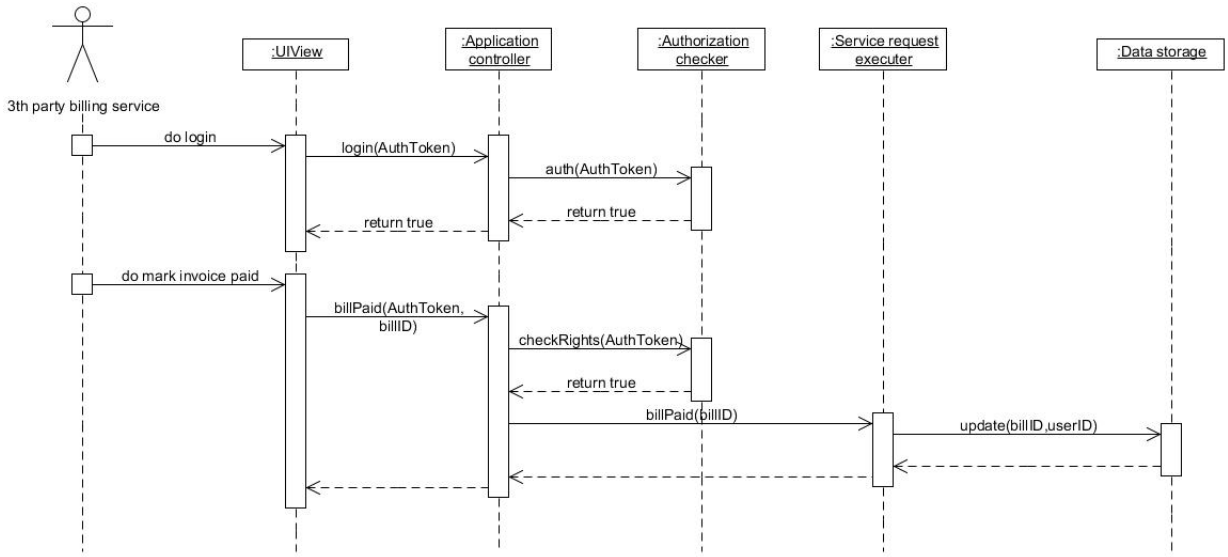

Figure 38: The sequence diagram for receiving a confirmation that a bill was paid.

Finally, this scenario is again very similar to the Alarm notification recipient configuration scenario. This action is available through the Application controller which provides an interface that the 3rd party billing service can use to notify ReMeS that a certain bill is paid.

10 Understanding the architecture

In this section a usage scenario in the provided ReMeS system will be explained by using the ReMeS components and client-server view diagrams. The scenario in question is the scenario where an alarm arrives at the ReMeS back-end and the actuator is activated.

When an alarm trame arrives at the ReMeS back-end, first of all the incoming communication component will translate the stream of bits to a NativeDataTrame. Then the same component will analyze that NativeDataTrame to find out what kind of trame it is. This analyzation will conclude that it is indeed an alarm trame.

Since the trame is an alarm trame, the incoming communication component will call the method receiveAlarmTrame in the alarm processor. The alarm processor will store this alarm in the database and will receive the alarm configuration data from the database. If this data includes a customer notification, it will call the notifyAlarm method of the outgoing communication component. This outgoing communication component will then make sure that the customer gets notified about the alarm.

If the alarm configuration data includes a valve actuation, the actuator controller will be called by the alarm processor. The actuator controller will then create an actuator message for the specific valve and call the correct method in the outgoing communication component. If the incoming communication component receives an acknowledgement, this data will be stored in the database. If the incoming communication component doesn't receive an acknowledgement trame, the actuator controller will make the outgoing communication component resend the control trame. The actuator controller will attempt to do this a certain amount of times. If there is still no acknowledgement received, the notifyIssue method will be called in the outgoing communication component. The customer will be notified about the issue.

11 Privacy analysis

11.1 Data Flow Diagram

The data flow diagrams (DFD) depicted in figures 39 and 40 are based on the client-server view of the ReMeS system.

This section should contain the DFD diagram + an explanation of the decisions you made.

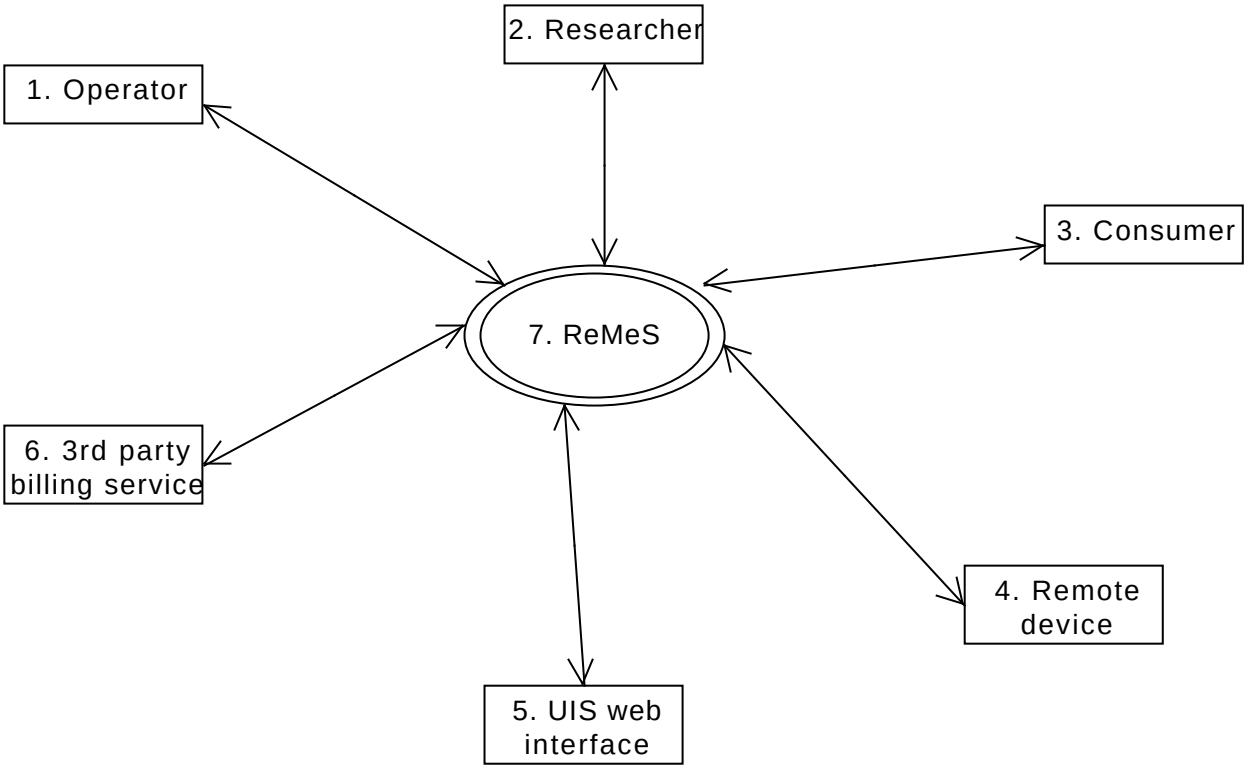

Figure 39: The Level 0 DFD diagram for ReMeS.

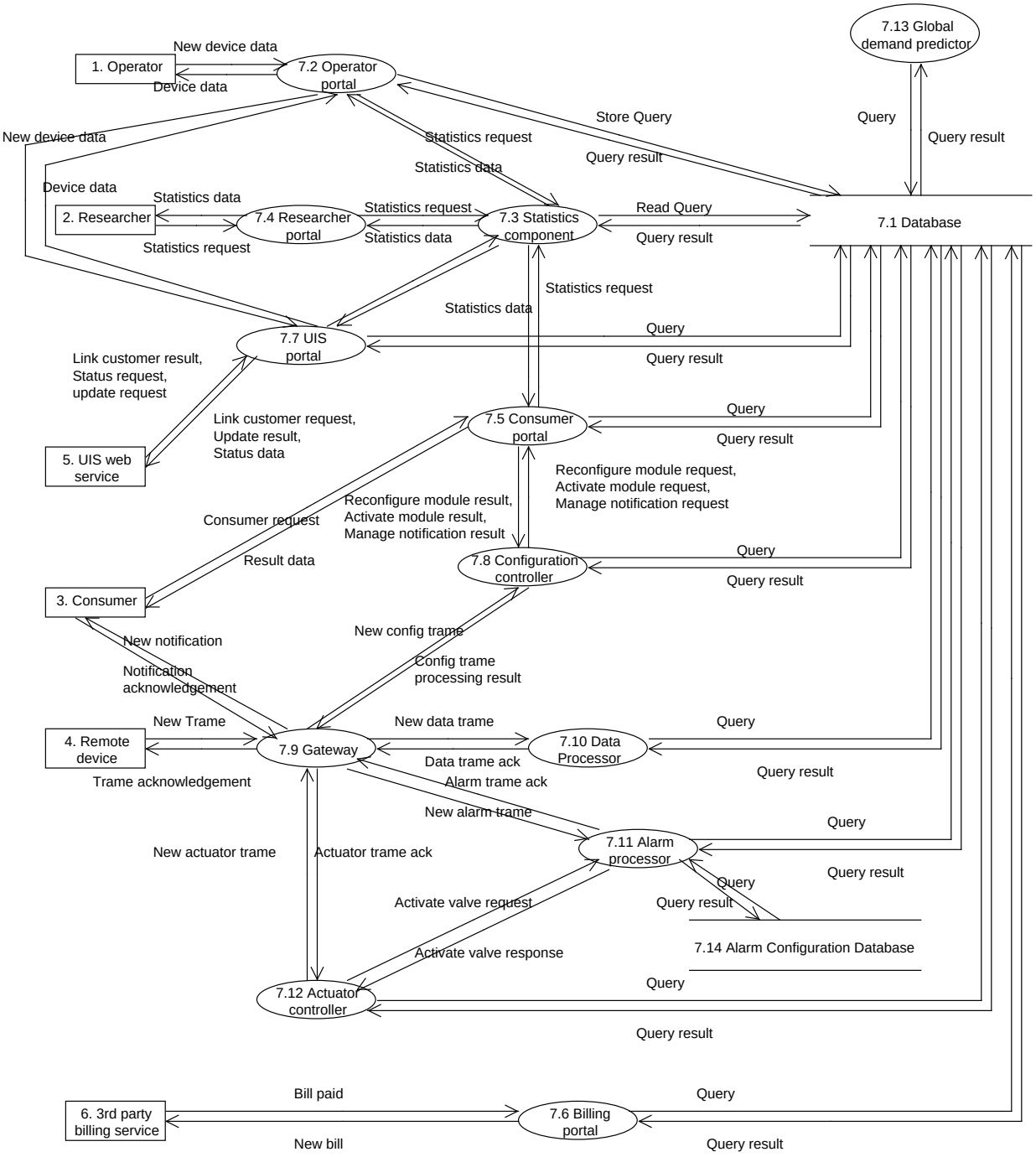

Figure 40: The Level 1 DFD diagram for ReMeS.

The DFD slightly varies from the original client-server view, as the following decisions were made:

1. All frontend components were decomposed into an external entity and the actual process.
2. The price rate notifier is not depicted in the client-server view, and is assumed to be part of the UIS web component.
3. The UIS web service entity represents the UIS component that actually uses the ReMeS interface to communicate and not the offered web service to the UIS component.

4. The authentication component is missing from the diagrams because it is assumed that the session token is sent from the users to the portal for each request.
5. The Data processor component and the anomaly detector component were combined to one process: Data processor, because the additional process would not introduce any more information about or threats to the system.
6. The *Store query* represents an insert request to the database of some sorts, while the *Read query* represents a retrieval query for the database. If just *Query* is used, it is assumed that the flow will contain both read and insert requests.

11.2 Mapping of threats to DFD

This section gives an overview of the different threats that exist for each DFD element. The DFD element names are represented in the DFD of the previous section! This table is split up in to different sections for data stores, data flows, processes and entities.

	Threat target	L	I	N	D	D	U	N
Data Store	Database (7.1)	×	×	×	×	×		×
	Alarm configuration DB (7.14)	×	×	×	×	×		×
Data Flow	Operator – Operator portal (1 - 7.2)	×	×	×	×	×		×
	Operator portal – Operator (7.2 - 1)	×	×	×	×	×		×
	Researcher – Reasearcher portal (2 - 7.4)	×	×	×	×	×		×
	Researcher portal – Reasearcher (7.4 - 2)	×	×	×	×	×		×
	Consumer – Consumer portal (3 - 7.5)	×	×	×	×	×		×
	Consumer portal – Consumer (7.5 - 3)	×	×	×	×	×		×
	Consumer – Gateway (3 - 7.9)	×	×	×	×	×		×
	Gateway – Consumer (7.9 - 3)	×	×	×	×	×		×
	Remote device – Gateway (4 - 7.9)	×	×	×	×	×		×
	Gateway – Remote device (7.9 - 4)	×	×	×	×	×		×
	UIS web service – UIS portal (5 - 7.7)	×	×	×	×	×		×
	UIS portal – UIS web service (7.7 - 5)	×	×	×	×	×		×
	3rd party billing service – Billing portal (6 - 7.6)	×	×	×	×	×		×
	Billing portal – 3rd party billing service (7.6 - 6)	×	×	×	×	×		×
	Operator portal – UIS Web service (7.2 - 7.7)	×	×	×	×	×		×
	UIS Web service – Operator portal (7.7 - 7.2)	×	×	×	×	×		×
	Operator portal – Database (7.2 - 7.1)	×	×	×	×	×		×
	Database – Operator portal (7.1 - 7.2)	×	×	×	×	×		×
	Operator portal – Statistics component (7.2 - 7.3)	×	×	×	×	×		×
	Statistics component – Operator portal (7.3 - 7.2)	×	×	×	×	×		×
	Researcher portal – Statistics component (7.4 - 7.3)	×	×	×	×	×		×
	Statistics component – Researcher portal (7.3 - 7.4)	×	×	×	×	×		×
	UIS portal – Statistics component (7.7 - 7.3)	×	×	×	×	×		×
	Statistics component – UIS portal (7.3 - 7.7)	×	×	×	×	×		×
	UIS portal – Database (7.7 - 7.1)	×	×	×	×	×		×
	Database – UIS portal (7.1 - 7.7)	×	×	×	×	×		×
	Consumer portal – Statistics component (7.5 - 7.3)	×	×	×	×	×		×
	Statistics component – Consumer portal (7.3 - 7.5)	×	×	×	×	×		×
	Consumer portal – Database (7.5 - 7.1)	×	×	×	×	×		×
	Database – Consumer portal (7.1 - 7.5)	×	×	×	×	×		×
	Statistics component – Database (7.3 - 7.1)	×	×	×	×	×		×
	Database – Statistics component (7.1 - 7.3)	×	×	×	×	×		×
	Configuration controller – Consumer portal (7.8 -7.5)	×	×	×	×	×		×
	Consumer portal – Configuration controller (7.5 -7.8)	×	×	×	×	×		×
	Configuration controller – Database (7.8 - 7.1)	×	×	×	×	×		×
	Database – Configuration controller (7.1 - 7.8)	×	×	×	×	×		×

	Gateway – Configuration controller (7.9 - 7.8)	×	×	×	×	×	×
	Configuration controller – Gateway (7.8 - 7.9)	×	×	×	×	×	×
	Gateway – Data processor (7.9 - 7.10)	×	×	×	×	×	×
	Data processor – Gateway (7.10 - 7.9)	×	×	×	×	×	×
	Data processor – Database (7.10 - 7.1)	×	×	×	×	×	×
	Database – Data processor (7.1 - 7.10)	×	×	×	×	×	×
	Gateway – Alarm processor (7.9 - 7.11)	×	×	×	×	×	×
	Alarm processor – Gateway (7.11 - 7.9)	×	×	×	×	×	×
	Alarm processor – Alarm Configuration Database (7.11 - 7.14)	×	×	×	×	×	×
	Alarm Configuration Database – Alarm processor (7.14 - 7.11)	×	×	×	×	×	×
	Alarm processor – Database (7.11 - 7.1)	×	×	×	×	×	×
	Database – Alarm processor (7.1 - 7.11)	×	×	×	×	×	×
	Alarm processor – Actuator controller (7.11 - 7.12)	×	×	×	×	×	×
	Actuator controller – Alarm processor (7.12 - 7.11)	×	×	×	×	×	×
	Gateway – Actuator controller (7.9 - 7.12)	×	×	×	×	×	×
	Actuator controller – Gateway (7.12 - 7.9)	×	×	×	×	×	×
	Actuator controller – Database (7.12 - 7.1)	×	×	×	×	×	×
	Database – Actuator controller (7.1 - 7.12)	×	×	×	×	×	×
	Billing portal – Database (7.6 - 7.1)	×	×	×	×	×	×
	Database – Billing portal (7.1 - 7.6)	×	×	×	×	×	×
	Global demand predictor – Database (7.13 - 7.1)	×	×	×	×	×	×
	Database – Global demand predictor (7.1 - 7.13)	×	×	×	×	×	×
Process	Operator portal (7.2)	×	×	×	×	×	×
	Researcher portal (7.4)	×	×	×	×	×	×
	Actuator controller (7.12)	×	×	×	×	×	×
	Consumer portal (7.5)	×	×	×	×	×	×
	Billing portal (7.6)	×	×	×	×	×	×
	UIS portal (7.7)	×	×	×	×	×	×
	Statistics component (7.3)	×	×	×	×	×	×
	Alarm Processor (7.11)	×	×	×	×	×	×
	Data processor (7.10)	×	×	×	×	×	×
	Configuration controller (7.8)	×	×	×	×	×	×
	Global demand predictor (7.13)	×	×	×	×	×	×
Entity	Operator (1)	×	×			×	
	Researcher (2)	×	×			×	
	Consumer (3)	×	×			×	
	Remote Module (4)	×	×			×	
	UIS web service (5)	×	×			×	
	3rd party billing service (6)	×	×			×	

Table 1

After making different assumptions about the system and analysing the possible threats further, a lot of possible threats were ruled out. The final threat table can be seen in the following figure. In this table only the actually handled threats are represented.

	Threat target	L	I	N	D	D	U	N
Data Store	General Database (7.1)	×	×			×		×
Data Flow	Operator – Operator portal (1 - 7.2)							×
	Researcher – Reasearcher portal (2 - 7.4)							×
	Consumer – Consumer portal (3 - 7.5)							×
	Consumer – Gateway (3 - 7.9)							×
	Remote device – Gateway (4 - 7.9)							×
	UIS web service – UIS portal (5 - 7.7)	×	×					×
	3rd party billing service – Billing portal (6 - 7.6)							×
	General internal dataflow		×			×		×
Process	General internal process (7.2)							×
Entity	Operator (1)							
	Researcher (2)							
	Consumer (3)		×				×	
	Remote Module (4)		×					
	UIS web service (5)							
	3rd party billing service (6)							

Table 2

11.3 Threat elicitation

11.3.1 Assumptions

This section discusses the different assumptions that were made. This includes general assumptions about the system (e.g. the data flow between X and Y is considered encrypted), and decisions and observations (e.g. Non repudiation is not considered a threat for this system). The reasoning behind each assumption and decision is also included.

These numbered assumptions will also be referenced to quite a lot in the threat elicitation section.

1. All internal processes can only be compromised by internal threats. We consider the back-end capable of protecting these internal processes from outsider threats. Therefor we will consider only one process, representing all internal processes, since the same threats apply to all of them.
2. Likewise to the previous assumption, all data flows between the internal processes are also considered as one data flow, since those threats apply to all of them. Like the previous assumption, data flows between internal processes are only vulnerable to internal threats.
3. Other dataflows (not between internal processes) are not grouped to one dataflow, since different threats may apply. Since these flows are not only between internal processes, the back-end cannot guarantee protection of these flows.
4. We assume that the data stores are encrypted and have a layer of access control installed. This leads us to trusting the data stores and as such, we will only consider one data store to represent all the stores. The data stores also have access control installed. This access control ensures that certain processes/entities only have access to the bare minimum amount of information needed for that process/entity. Queries requesting more information than a process or entity is allowed to have are blocked by this control. Since the data stores are encrypted very well and there is access control installed, identifiability and linkability are not an issue with our data stores (except for the administrator who has all access rights).
5. No non-repudiation threats exist in our system. Our system stores utility usages and creates invoices based upon these usages. This means our system components do not need plausible deniability.
6. Detectability also is not a pressing concern in our system. The privacy concerns of this system are all focused on the data itself, not on the detectability of it.
7. Non-compliance is an important threat in our system. However, this threat is not specific for a component of our system, but poses to the system as a whole. We will therefor not make a distinction between the different DFD elements for this threat.
8. The only dataflow between entities and internal processes that is susceptible to linkability and identifiability threats is the data flow between the external UIS service and the UIS portal process. Linking the company contacted with internal data flows might reveal personal information of a customer such as his utility provider. Since we allready assumed that the internal components are shielded off from the outside world and their threats, we can assume that linkability and identifiability are not an issue for the internal data flows of our system.
9. We assume that the data flows between external entities and internal processes are SSL-encrypted. We will also assume that trames sent via sms/gprs are also using an encrypted channel, provided by the service provider.
10. The only entities that directly add data to the system are the consumer and the remote device. Therefor we can assume that content unawareness is not a threat to the other entities.
11. The remote devices are preprogrammed to only send useful data. We can assume that content unawareness is not a threat for the remote device entity.

12. The identifiability of the entities *consumer* is considered to be a threat. The other entities all use a unique identifier and have no possible need to hide their identities.
13. Information disclosure of data not meant to be accessed by certain internal users is not an issue because of the access control to the data stores, mentioned in assumption 4. Every user is able to access only specific views of the data store needed to execute their function.
14. We assume that the portals for employees are only accessible from inside the company network and not via the internet. This limits the threats to unauthorized access to customer or researcher functions from outside the system.
15. The authentication system is assume to be well implemented and secure.
16. The authentication credentials are also assumed to be stored securely on the server side.
17. Internal processes are not susceptible to corruption as we assume processes are implemented correctly and input is suficiently validated, and memory access is dealt with as well.
18. Side channel attacks on data flows are not considered as they are highly-unlikely to occur because they take a lot of analysis and the extracted information is not in correspondence of the effort.

11.3.2 Threats

This section documents the threats which were uncovered in the mapping section. Each threat description strictly follows the provided template.

T01 - Linking Alarm configuration data to user data

Summary:

The administrator or another insider with administrator rights is able to access both the alarm configuration database and the general database and will be able to link these two together. Access control to both these databases limit the type of requests a user can issue on any database but administrator rights are transitive to all other types of users.

Primary mis-actor:

Moderately skilled insider who is able to execute and manipulate his own access rights.

Basic path:

bf1. The misactor invokes the rights of a user with valid access to the alarm configuration database

bf2. The misactor retrieves information from the alarm configuration database

bf3. The misactor invokes the rights of a user with valid access to the general database

bf4. The misactor retrieves information from the general database

bf5. The misactor links both sets of data (e.g. based on a shared foreign key)

Consequence:

The combined sets of data contain possibly sensitive personal data such as emergency notification telephone numbers. This poses an actual privacy threat when the misactor sells this information to interested parties, such as criminal organisations.

Reference to threat tree node(s):

L_ds2, L_e6.

Parent threat tree(s):

L_ds, L_e.

DFD element(s):

7.14 Alarm configuration database, 7.1 Database

Remarks:

r1. The L_ds1 requirement of weak access is only fulfilled when the misactor has the access rights of the root of the access right hierarchy. In all other instances this precondition is not met due to the installed access control for regular users. These root users will have the ability to circumvent the installed access control in some particular situations.

r2. The linkability of entity leaf node L_e6, indicating linkability based on a device's temporary ID, inspired to this data store linkability threat.

T02 - Information disclosure of customer usage history

Summary:

An authenticated internal user can access personal usage history of any customer he wants.

Primary mis-actor:

Unskilled insider

Basic path:

bf1. The misactor authenticates himself (by using his own credentials or by spoofing another valid user).

bf2. The misactor gains indirect access to the database and is able to perform queries about certain certain contents.

Consequence:

Confidential usage history data and personal data is exposed to the authenticated user (researcher or operator). This way an unauthorised user can access information that is not supposed to be available to said user.

Reference to threat tree node(s):

ID_ds9, ID_ds2

Parent threat tree(s):

ID_ds

DFD element(s):

7.1 Database

Remarks:

r1. This threat is only applicable to the general database (7.1) because the alarm configuration database is not accessible via non automated entities.

r2. Spoofing a user is considered a precondition of this threat. By spoofing another user, different contents in the database is made accessible to the user.

r3. Although access control is installed for the data stores as explained in assumption 4 and that only internal users (e.g.employees) have access to the data stores (assumption 14, the protection scheme can be circumvented as specified in ID_ds9 by internal employees spoofing other types of employees and thus gaining other access rights. See threat T03.

r4. We assume the data store itself is sufficiently protected which eliminates unencrypted data, side channel attacks (ID_ds4), extra-monitor access (ID_ds3) and bad storage management (ID_ds5) (assumption 4).

T03 - Spoofing an internal user of ReMeS by falsifying credentials

Summary:

The misactor obtains user credentials allowing him to log in and access the system as a user other than himself.

Primary mis-actor:

Skilled insider

Basic path:

bf1. The misactor gains access to the credentials of a user (by stealing, guessing, etc.)(S_8, S_12, S_13)

bf2. The misactor uses the credentials to log in to the system as another type of user.

bf3. The misactor receives all priveleges of the spoofed employee

Consequence:

Confidential data can possibly be viewed by users that are normally not able to view this data. This misactor could possibly expose sensitive data to the outside.

Reference to threat tree node(s):

S_8, S_12, S_13

Parent threat tree(s):

ID_ds, S

DFD element(s):

1. Operator, 2. Researcher

Remarks:

r1. An authentication system is present in the architecture, which rules out threat S_4.

r2. The authentication process is considered secure (assumption 15) thus the tampering threat (leaf of S_3) does not hold, and it does not support null credentials (S_10) or equivalence (S_09), downgrade authentication (S_11) or weak change management (S_09). Also no key distribution storage is present (S_14).

r3. Spoofing due to weak server-side storage (S_15) is not possible because of assumption 16.

r4. Spoofing due to weak transit is not possible due to the strong ssl encryption of the communication between entities and portals (assumption 9).

r5. Spoofing only applies to operators and researchers (assumption 14).

r6. Spoofing due to weak credential storage on client-side is described in T04 - Spoofing a user of ReMeS because of weak credential storage. .

T04 - Spoofing a user of ReMeS because of weak credential storage.

Summary:

The misactor obtains user credentials from a client side system allowing him to log in and access the system.

Primary mis-actor:

Skilled insider or skilled outsider

Basic path:

bf1. The misactor gains access to the credentials of a user by weak credential storage at the client side (e.g. unprotected text file for easy recall of credentials)(S_13).

bf2. The misactor uses the authentic credentials to log in to the system as an other user than himself.

bf3. The misactor receives all privileges of the spoofed employee or customer

Consequence:

Confidential data is possibly exposed to outsiders.

Reference to threat tree node(s):

S_8, S_12, S_13

Parent threat tree(s):

ID_ds, S

DFD element(s):

1. Operator, 2. Researcher, 3.Consumer

Remarks:

r1. An authentication system is present in the architecture, which rules out threat S_4.

r2. The authentication process is considered secure (assumption 15) thus the tampering threat (leaf of S_3) does not hold, and it does not support null credentials (S_10) or equivalence (S_09), downgrade authentication (S_11) or weak change management (S_09). Also no key distribution storage is present (S_14).

r3. Spoofing due to falsifying crednetials is described in T03 - Spoofing an internal user of ReMeS by falsifying credentials.

r4. This form of spoofing can apply to outside users (consumers) or inside users (operators and researchers), but spoofing outside users does not affect the inside entities or vice versa.

T05 - Linkability of requests sent to external UIS

Summary:

The misactor links several requests to the same customer and creates a profile of this customer.

Primary mis-actor:

unskilled insider (external UIS) / skilled outsider

Basic path:

bf1. A bill is being made for a certain customer. Requests are sent to the correct UIS (there exist different companies, requests need to be sent to the UIS of the company who has the customer).

bf2. The misactor intercepts the data flow.

bf3. The misactor can link several requests to the same customer.

Consequence:

The misactor can build a profile of the patient. The misactor knows the utility company of the customer.

Reference to threat tree node(s):

L_df1, L_df8

Parent threat tree(s):

L_df

DFD element(s):

data flow from UIS portal to UIS web service (7.7 - 5).

Remarks:

r1. The misactor doesn't have to know about the content being sent, only which UIS is being consulted.

r2. To link the different requests, the misactor has to know for which customer a bill is being made, which makes this threat very unlikely.

r3. The right branch of the tree (insecure anonymity system (L_df4)) and the other leaf nodes of the non-anonymous communication branch (L_df3) are not considered, as it is not the sender (browse service) whose identity should be protected, but the patient, who is not directly part of the data flow.

T06 - Information disclosure internal process

Summary:

The misactor gains access to one of the internal processes.

Primary mis-actor:

Authorized insider

Basic path:

bf1. The misactor has the required privileges to access to processes.

bf2. The misactor uses his privileges to access information outside the scope of his job.

Consequence:

The misactor has access to (possibly sensitive) personal identifiable information.

Reference to threat tree node(s):

ID_p

Parent threat tree(s):

ID_p

DFD element(s):

All internal processes (7.2 - 7.13)

Remarks:

r1. This threat especially applies to administrators of our system. Our databases have access control installed, but administrators will still have full access to all the data (assumption 4).

r2. This threat is inspired by *spoofing* an entity leaf threat, however, when an insider has to much privileges, this threat applies as well. Spoofing entities with access to internal processes is not considered, as we assume the system is physically protected(assumption 1).

r3. We assume processes are not corruptable (assumption 17).

r4. The side channel attack is described in T07 - Side channel information disclosure internal process.

T07 - Side channel information disclosure internal process

Summary:

The misactor gains access to one of the internal processes.

Primary mis-actor:

skilled insider

Basic path:

bf1. The misactor performs a side channel attack on one of the internal processes.

bf2. The misactor obtains process information.

Consequence:

The misactor obtains process information.

Reference to threat tree node(s):

ID_p2

Parent threat tree(s):

ID_p

DFD element(s):

All internal processes (7.2 - 7.13)

Remarks:

r1. The alternative spoofing attack is described in T06 - Information disclosure internal process.

r2. We assume processes are not corruptable (assumption 17).

T08 - Non-compliance of employees

Summary:

The ReMeS operators don't process the customer data in compliance with legislations or policies.

Primary mis-actor:

insider (operators, ...).

Basic path:

bf1. The misactor fails to comply with the community's policy or legislation (e.g. the customer's data is revealed to third parties).

Consequence:

The customer's personal information is shared without his knowledge. When detected, ReMeS can get fined, and its trustworthy reputation is ruined.

Reference to threat tree node(s):

PN_2

Parent threat tree(s):

PN

DFD element(s):

all (except entities)

Remarks:

r1. This threat applies to the entire system, as no individual DFD element is specifically targetted (assumption 7).

r2. A similar threat which is posed by the developer is described in T10 - Non-compliance management.

r3. A specific non-compliance threat concerning cnsents is described in T09 - Missing user consents.

T09 - Missing user consents

Summary:

The system did not ask the customer's permission to share part of his (pseudonymized) personal information with 3rd parties.

Primary mis-actor:

Management

Basic path:

bf1. The management fails to require customer consents to be included in the user flow.

bf2. The user is unable to state his preferences concerning personal data sharing.

Consequence:

The customer's personal information will be shared with 3rd parties against his will.

Reference to threat tree node(s):

PN_3

Parent threat tree(s):

PN

DFD element(s):

entire system (except entities)

Remarks:

r1. This threat applies to the entire system (assumption 7.

r2. Two general threats which correspond to general non-compliance are described in T08 - Non-compliance of employees and T10 - Non-compliance management.

T10 - Non-compliance management

Summary:

The management fails to request a design and implementation of the system in compliance with legislation.

Primary mis-actor:

Management

Basic path:

bf1. The misactor fails to require a system that is legally compliant (either he is unaware of the legislation or he consciously decides to ignore it).

bf2. The customer data is not processed or collected in accordance to (privacy) legislation.

Consequence:

The customer's personal information is shared without his knowledge. When detected, ReMeS can get fined, and its trustworthy reputation is ruined.

Reference to threat tree node(s):

PN_2

Parent threat tree(s):

PN

DFD element(s):

all (except entities)

Remarks:

r1. This threat applies to the entire system, as no individual DFD element is specifically targetted.

r2. A similar threat which is posed by the employees when the system is up-and-running is described in T08 - Non-compliance of employees.

r3. A specific non-compliance threat concerning cnsents is described in T09 - Missing user consents.

T11 - User unawareness

Summary:

The user is unaware of the consequences of sharing information (e.g. by sharing too much information even anonymized data can reveal the user's identity).

Primary mis-actor:

Management

Basic path:

bf1. The management fails to add as requirement the need of notifications and warnings when the patients intends to upload sensitive and/or identifiable content.

bf2. The user adds information to the system which can easily identify him (e.g. a picture of himself) as he is unaware of the consequences.

Consequence:

When ReMeS processes retreive information, the identifiable information is returned. The user's privacy is thus violated as he assumes that his information stays confidential and his identity will not be revealed.

Reference to threat tree node(s):

U_1

Parent threat tree(s):

U

DFD element(s):

3. consumer

Remarks:

r1. This threat only applies to the consumer (assumption 10 and 11).

r2. The threat concerning inaccurate user information is described in T12 - content inaccuracy.

T12 - content inaccuracy

Summary:

The customer failed to update his administrative information

Primary mis-actor:

Management

Basic path:

bf1. The management fails to indicate the need of a notification that warns the user of the importance of up-to-date and accurate information.

bf2. The customer provides inaccurate or incomplete administrative information or fails to update old information.

Consequence:

The customer himself or ReMeS processes consult the customer's inaccurate administrative information and create incorrect data.

Reference to threat tree node(s):

U_3, U_4

Parent threat tree(s):

U

DFD element(s):

3. consumer

Remarks:

r1. This threat only applies to the customer (assumption 10 and 11).

r2. The threat concerning users providing too much information is described in T11 - User unawareness.

11.4 Prioritization of threats

This section provides a list of the threats (ID + title) of the previous section. The order is based on the threat's risk (likelihood * impact). The distinction was made between high, medium, and low risk and, within each category, the threats are also ordered according to their risk. Finally, an explanation will be given of why the threats were ordered in this particular way.

11.4.1 High priority

The threat that was given the highest priority is T02 - Information disclosure of customer usage history. The main reason for this choice is the fact that is one of the more significant threats to the privacy of the customers. This threat is also very difficult to deal with. Other threats that were labeled as high priority threats include the threats about spoofing an internal user of ReMeS. Motivated attackers wishing to exploit this threat could cause ReMeS harm in name and function. This should, evidently, be avoided at all cost.

- T02 - Information disclosure of customer usage history
- T03 - Spoofing an internal user of ReMeS by falsifying credentials
- T04 - Spoofing a user of ReMeS because of weak credential storage.

11.4.2 Medium priority

The threats that were labeled as being of medium priority are placed in this category because of the fact that some form of trust has been put in the employees of ReMeS. Misuse of this trust will have serious consequences for the misactors and this should be incentive enough to mitigate the threat for these specific cases. Other than that, the medium priority threats could be mitigated further or avoided altogether without too much trouble from the ReMeS side.

- T11 - User unawareness
- T09 - Missing user consents
- T01 - Linking Alarm configuration data to user data
- T10 - Non-compliance management
- T12 - content inaccuracy

11.4.3 Low priority

Due to the fact that some level of trust is put in our employees, some threats are of a low priority. The list below contains threats that are just rather unlikely to occur because of the nature of the consequences for the misactors or the fact that the trouble that a misactor has to go through to exploit such a threat is not worth the merit of the exploit. (e.g. side channel information disclosure of internal processes).

- T08 - Non-compliance of employees
- T07 - Side channel information disclosure internal process
- T06 - Information disclosure internal process
- T05 - Linkability of requests sent to external UIS

12 Appendix

12.1 Element Catalog

12.1.1 Incoming gateway

The incoming gateway is the entry point for trames sent by a remote device, in the ReMeS system. This module dispatches the incoming trames to the components in the system that are fit to handle them. The incoming gateway also handles the incoming acknowledgement trames and provide a callback event for components waiting on such an acknowledgement.

Message router

The message router is the subcomponent that handles the dispatching of the trames to the modules that are capable of processing them.

Acknowledgement handler

If the type of trame is an acknowledgement trame, the acknowledgement handler will process this incoming acknowledgement. Also, other incoming trames are passed to the acknowledgement to make sure those incoming trames are acknowledged by ReMeS itself.

12.1.2 Scheduler for incoming alarm trames

The Scheduler for incoming alarm trames handles, as the name indicates, the incoming alarm trames. These trames are temporarily stored in a prioritized queue, which makes it easy to prioritize incoming trames. The scheduler component than empties the queue at as fast a pace he can manage and dispatches the commands to send an emergency control trame and/or send a notification to the alarm recipients.

Trame handler

A proxy instance which handles the external interface of this component and fills up the priority queue.

Input router

The input router component is a subcomponent visible in the decomposed view of the trame handler. This component handles the passing of incoming trame messages to the next supercomponent (in this case being the buffer). While passing the regular trame, it also stores part of the trame with the trame header (containing the time stamp, device id and such) in the latest heartbeat table.

Latest heartbeat table

The latest heartbeat table is a subcomponent of the trame handler component. This component stores incoming measurement trame headers containing time stamp information and device id's. This table is used to monitor the incoming measurements and possibly detect missing measurements.

Active loop

The active loop component is a subcomponent of the trame handler. This components gets its name from the fact that this component works in a continuous loop in it's own thread of control. This component iterates over the entries in the latest heartbeat table and monitors for missing measurements. This component also queries the data storage unit for the measurement schedules of the remote devices. These are needed to check for missing measurements.

Buffer

This is a buffer in which data is stored based on priority. The implementation of this element could be achieved by using a priority queue implementation.

Buffer frontend

The buffer frontend component is a subcomponent of the buffer component. The components in this decomposed view illustrate the redundancy measures that have been taken to effect the availability quality attribute scenarios. The Buffer frontend is an simple component offering an external interface and delegating the requests to add an element to each buffer instance. For retrieving data, a request is delegated to the buffer instances and the first response is used.

Buffer implementation

The buffer implementation component is a child component of the buffer component. This component represents the actual physical buffer. The instances manifesting this component will store and hold the actual data. These components will use the same interface as the previously discussed buffer frontend component.

Scheduler

This scheduler empties the queue and handles the incoming trames based on priority. This scheduler handles this responsibility with fairness, such that eventually all trames will be handled. This element will use a dynamic scheduling policy to avoid starvation.

12.1.3 Scheduler for outgoing trames

The scheduler for outgoing trames handle the requests for trames that need to be sent out from the ReMeS system. This component also uses a prioritized queue to sort and schedule those requests based on priority. This module has the capability of querying the central data storage unit for information on how a recipient needs to receive a trame or through which network technology.

Request handler

The request handler is a proxy for offering the external interface to this subcomponent. All received requests are stored into the prioritized queue.

Buffer

The prioritized queue is the same component as previously discussed in section 12.1.3.

Trame constructor

The trame constructor has the responsibility to empty the queue and process the incoming requests in such a manner that they are processed fairly, but based on their priorities. This component can also query the data storage unit for information on the recipient. This information will contain the technology that needs to be used to transfer the trame and the trame constructor will construct and prepare the outgoing trames in a manner compatible with that technology.

12.1.4 Outgoing gateway

The outgoing gateway component handles the physical dispatching of outgoing messages. These messages can be trames, but also email messages are possible. This gateway has the capability of sending those messages using a number of different technologies such as GPRS/3G, sms or just plain over tcp/ip.

Request handler

The request handler serves as a proxy for offering the external interface of this component. This proxy dispatches the outgoing messages based on the type of network technology that needs to be used.

Buffer

For each network technology a buffer exists which holds the yet to be sent messages.

Message constructor

For each network technology there exists a message constructor component which handles the construction of messages or packets for that specific network technology. It sort of wraps the data that has to be sent with the adequate packet data that the network can accept.

Network broker

For each network technology, there is a network broker component in place which handles the effective communication with the network. It is this component that will handle the final dispatching of messages over the network.

12.1.5 User notification

The user notification component handles the request for notifying users. This component also has the capability of polling the data storage unit for information about the user that needs to be notified.

Request handler

The request handler is a proxy for providing an external interface for the internal functionality.

Buffer

This buffer is a simple component for temporary storing the requests until they can be handled.

Notification constructor

The notification constructor handles the construction of the notification messages (email structure for example). These notification messages form the actual content that will be provided to the outgoing gateway.

12.1.6 Scheduler for incoming measurement trames

This component handles the scheduling of incoming measurement trames to the measurement processing service. When a measurement trame arrives, it will also be checked if the trame arrives in time or if a scheduled check in has been missed. This scheduling component also monitors the event channel for information on the load of other heavy duty components and considers this when scheduling new requests to the data storage unit (for example).

Trame handler

The trame handler functions as the entry point for this scheduler component and handles the dispatch to the buffer used for final scheduling. In this trame handler the regularity of the device, when it comes to sending measurement updates, is checked against the intended check in schedule. The moment a device sends a measurement trame, it is entered in the heartbeat table. The active loop request the device configuration from the data storage unit and uses that schedule to detect missing check ins.

Buffer

This buffer is a very important buffer in the system. In case of a database failure, this buffer will hold the measurement trames until the other components return to normal operation.

Event channel

The event channel in this component is the event channel used throughout the ReMeS system. This component is used to publish notifications to and to receive notifications from about the state of load on the system.

Scheduler

This scheduler component, schedules in a fair manner, the measurement trames to the data storage unit for processing. This component uses the event channel to monitor the operation mode of some heavy duty components.

12.1.7 Watchdog

The watchdog component signifies the capability of the system to monitor the operational state of different components. Different important components in the system can post heartbeat updates to this watchdog component. This component handles the monitoring and storage of these heartbeats.

Active loop

The active loops continuously checks the heartbeat table for new entries. If an anomaly is found, it posts a message in the event channel.

Event channel

This event channel component is the same component as the other event channels.

Heartbeat table

This heartbeat table registers and stores the heartbeats until the active loop can process data from them.

12.1.8 Invoice manager

The invoice manager component handles the creation and dispatching of invoices to the third party billing service. The creation of invoices can be triggered by synchronous and asynchronous calls to the invoice generation.

UIS communicator

The UIS communicator will handle the communication with the Utility Information System. This component will use the external interface provided by the Utility Information System. The details of how this communication happens, depend on the interface provided by the UIS and is outside the scope of this assignment.

3rd party billing service communicator

The 3rd party billing service communicator component handles the communication with the 3rd party billing service for sending them the generated invoices. The same limitations apply as in the previously discussed section.

Invoice generator

The invoice generator handles the responsibility of creating the invoices based on the stored consumption history and measurement data. The generation of these invoices is triggered either by direct method call or by a notification in the event channel.

Event channel

The event channel component has been amply discussed in other components throughout the system.

12.1.9 Computation of consumption prediction

The computation of consumption prediction component represents the internal component that will actually be computing the consumption predictions.

Consumption prediction algorithm

The consumption prediction algorithm component represents an algorithm to be used for performing consumption prediction. This component however is not fully decomposed or designed to specification due to timing constraints. A strategy pattern could be used to organise this component further if the need arises.

12.1.10 Scheduler for consumption prediction requests

This scheduler component handles the scheduling of consumption prediction data. The structure of this component is fairly similar to previously discussed scheduling components. The elements responsible for initiating requests is primarily the user interaction component discussed later on.

Request handler

This component is a proxy which provides an external interface to the internal components. This component stores the requests for consumption prediction in a temporary buffer.

Buffer

This buffer stores the request before they can be scheduled and handled for processing.

Scheduler

The scheduler actually schedules and triggers the computation of consumption predictions for the ReMeS system.

12.1.11 User interaction

The user interaction component handles the interaction of different types of users, with the system. This module represents the user interface and the handling of user initiated requests for data processing or other stuff. This form of user interaction makes use of the MVC pattern as described in earlier sections.

Application controller

The application controller represents an application layer for handling the communication between the user interface controls and views and the core of the ReMeS system. External applications and user interfaces can make requests to this controller layer so they don't need to interact with the core itself.

UIView

The UIView component is an component representing the effective user interface that will be represented to clients of the ReMeS system. This user interface is related to the view component in an MVC architecture. It only directly communicates with the controller layer which is, in this case, represented by the Application controller.

Service request executer

The Service request executor component delegates service requests to components in the core of the system.

Authorization checker

The Authorization checker is a module that handles the authorisation of users which are using the application controller or the user interface directly. This component has the responsibility of providing authentication services when needed.

Event channel

This event channel has been discussed a few times already. One extra function for usage in this components can be found in the fact that some operator user interfaces use this event channel to monitor the uptime states of different components. If a component of some sort fails and this failure is detected, this event will be communicated to an operator interface through this event channel.

12.1.12 Data storage

The data storage component represents the central data repository that will be used by a lot of components in the ReMeS System.

DB request handler

The DB request handler is a proxy for the database functionality representing a unified external interface to the database functionality. This handler component internally uses a caching module, which improves performance and availability within given boundaries. This component is also capable of delegating queries to the database instances that are capable of handling them.

Request handler

The request handler component is a child component of the DB request handler component. This component handles the same functionality of the DB request handler, without the functionality added by decomposing said element any further. This request handler offers the external interface to the outside components to shield the implementation details from the rest of the system.

Buffer

This buffer component is a subcomponent of the DB request handler component. This component is nothing more than a simple buffer instance for storing requests before they can be handled.

Request cache

The Request cache component is a subcomponent of the DB request handler. This component handles the requests to the database and temporarily stores the results to offer a caching service.

When a request is pulled from the buffer, this cache component will first check if a result for this query already resides in the internal cache. If this is the case, and the result is fresh enough, the cache component will return this cached result without querying the database for the information. If an answer for the query is not present in the internal cache, or the version of the result is too old, the request will be passed along to the next component in line which will query the actual database for the results.

Parallel process scheduler

The parallel process scheduler is a subcomponent of the DB request handler component. This component schedules the incoming requests that couldn't be handled by the caching instance for actual processing. This component is capable of scheduling and dispatching multiple queries at once (hence the component name).

Measurement processor

The measurement processor accepts request from the request handler for incoming measurement requests and processes those measurements by extracting data, querying the anomaly detection unit on this data, and inserting the measurements at the correct location in the measurement database.

Anomaly detector

The anomaly detector actually runs various anomaly detection algorithms on the measurement data in order to try and detect anomalies in the consumption behavior of a certain user, based on reading of a certain device.

Event channel

The event channel used here has been amply discussed in previous sections.

User profile DB

This is the physical database component storing the actual user profiles.

Measurement DB

This is the physical database component storing the actual measurements

Remote device configuration DB

This is the physical database component storing the actual device configurations.

Invoice and billing DB

This is the physical database component storing the actual invoice and billing data.

12.2 Interface descriptions

12.2.1 IAcknowledgementHandler

Interface Identity

The only component that provides this interface is the Acknowledgement handler.

Resources Provided

- void acknowledge(Trame incomingTrame)
  This method should be called when for a certain incoming trame, an acknowledgement should be returned to the sender. This method will be used to send acknowledgements of measurement and alarm trames.
- void handleAck()
  This method should be called when an acknowledgement is received. The Acknowledgement handler will store this acknowledgement in the Data storage.

Data Type Definitions

- Trame
  The incoming trame that should be acknowledged.

Exception Definitions

12.2.2 IAnomalyDetector

Interface Identity

This interface is provided by the Anomaly Detector in the Data storage component.

Resources Provided

- void detectAnomaly(MeasurementData data)
  This method will invoke the anomaly detector to detect anomalies in the given data.

Data Type Definitions

- MeasurementData
  A set of measurement data needed to detect an anomaly.

Exception Definitions

12.2.3 IAuthenticator

Interface Identity

This interface is provided by the Authorization checker component.

Resources Provided

- boolean auth(AuthToken authToken)
  This method is used when a user wants to login into the Application controller. The method returns true if the authToken is valid.
- boolean checkRights(AuthToken authToken, Service service)
  This method is used by the Application controller to check wether or not a user is authorized to request a certain service. This method will return true if the user is authorized.

Data Type Definitions

- AuthToken
  The specific piece of information that can authenticate a user.
- Service
  A data type that represents a certain service of the User interaction component.

Exception Definitions

12.2.4 IBillSender

Interface Identity

This interface is provided by all components that can send an invoice to an external source. Currently these are the UIS communicator and the 3rd party billing service communicator components.

Resources Provided

- void sendInvoice(Invoice invoice)
  This method will send an invoice to the utility providing company.

Data Type Definitions

- Invoice
  A data type containing the full information of an invoice.

Exception Definitions

- CommunicationException
  This exception can be thrown if the component fails to establish a connection with the external source.

12.2.5 IBroker

Interface Identity

The IBroker interface is provided by all network brokers.

Resources Provided

- void send(Packet packet)
  Method to send a packet over a network (SMS, gprs, TCP/IP).

Data Type Definitions

- Packet
  A packet that can be sent over a network.

Exception Definitions

12.2.6 IBuffer

Interface Identity

This interface is used by all buffers in our system. This interface uses a generic type T.

Resources Provided

- void addToBuffer(T data)
  This method will add data to the buffer.
- T getNext()
  This method will get the next data from the buffer and return it.
- T getFromBuffer(Integer index)
  This method will get data from the buffer at a given position.

Data Type Definitions

- T
  This is a generic type.

Exception Definitions

- IndexOutOfBoundsException
  This exception can be thrown by the getFromBuffer(Integer index) method if there is no data at the given index.

12.2.7 ICompute

Interface Identity

This interface is provided by components who can compute a consumption prediction. At the moment the only component with this functionality is the Computation of consumption prediction component.

Resources Provided

- ConsumptionPrediction computePrediction(MeasurementData data)
  This method will request a consumption prediction with the given data. The method returns the prediction.

Data Type Definitions

- MeasurementData
  A set of measurement data needed to compute a consumption prediction.
- ConsumptionPrediction
  A computed consumption prediction.

Exception Definitions

12.2.8 IDataBase

Interface Identity

This interface is provided by the Data storage component. Internally, each separate database instance in the Data storage component also provides this interface.

Resources provided

- void create(CreateQuery q)
  This is one of the four basic database access methods provided. This method signifies the creation of a record based on a given query q.
- void update(UpdateQuery q)
  This is one of the four basic database access methods provided. This method signifies updating a record based on a given query q.
- DataTransferObject get(GetQuery q)
  This is one of the four basic database access methods provided. This method signifies retrieving a record based on a given query q. The information contained in this record will be returned as a DataTransferObject.
- void remove(RemoveQuery q)
  This is one of the four basic database access methods provided. This method signifies removing a record based on a given query q.

Data Type Definitions

- CreateQuery
  This data type represents a database query of the creation category. This CreateQuery type is a subtype of the general Query data type.
- UpdateQuery
  This data type represents a database query of the update category. This UpdateQuery type is a subtype of the general Query data type.
- GetQuery
  This data type represents a database query of the retrieval category. This GetQuery type is a subtype of the general Query data type.
- RemoveQuery
  This data type represents a database query of the remove category. This RemoveQuery type is a subtype of the general Query data type.
- DataTransferObject
  A data object that contains the information of a record in the Data storage.

Exception Definitions

- QueryFormatException
  This exception can be thrown when an argument representing a query to be executed, does not follow the expected formatting rules.
- DatabaseUnavailableException
  This exception can be thrown when the implementing component is unresponsive for any reason.

12.2.9 IExecuter

Interface Identity

This interface is provided by the Service request executer component. The methods of this interface are very similar to those of the IUserInteraction interface. All the methods are the same, except that they don't have an AuthToken. Every method of IUserInteraction is also provided by this interface (except for the login method). For further information we refer to section 12.2.22.

12.2.10 IHeartBeat

Interface Identity

This interface is provided by all heartbeat tables in the system.

Resources Provided

- updateHeartbeat(ID id)
  This method will update the latest heartbeat received of the given id to the current system time.

Data Type Definitions

- ID
  An identification id.

Exception Definitions

12.2.11 IHeartBeatTable

Interface Identity

This interface is provided by all heartbeat tables in the system.

Resources Provided

- Iterator iterate()
  A method to get the iterator of a heartbeat table. This iterator can be used to iterate over all elements in the heartbeat table and check if there are any that are overdue.

Data Type Definitions

- Iterator
  An iterator consisting of all elements found in a heartbeat table.

Exception Definitions

12.2.12 IInvoiceGenerator

Interface Identity

This interface is provided by components that can generate an invoice. Currently this is only the Invoice manager component.

Resources Provided

- Invoice generateInvoice(UserInfo userInfo)
  This method will generate an invoice for the user specified by the userInfo argument.

Data Type Definitions

- Invoice
  A data type containing the full information of an invoice.
- UserInfo
  Data type that contains information of a user.

Exception Definitions

- IncompleteDataException
  This exception can be thrown by generateInvoice(UserInfo userInfo). It is thrown when the UserInfo object does not contain sufficient data to correctly identify the customer.

12.2.13 IMeasProc

Interface Identity

This interface is provided by the Measurement processor.

Resources Provided

- void procNewMeas(Measurement meas)
  This method will feed new measurement data to the Measurement processor.

Data Type Definitions

- Measurement
  Data type containing measurement data.

Exception Definitions

12.2.14 IPredictionScheduler

Interface Identity

This interface is provided by the Scheduler for consumption prediction requests. This interface is different from IScheduler because it requires more information.

Resources Provided

- ConsumptionPrediction requestPrediction(PredictionType type)
  This method will schedule a consumption prediction. When the requested prediction is computed, this method will also return the prediction.

Data Type Definitions

- ConsumptionPrediction
  A computed consumption prediction.
- PredictionType
  The type of the requested consumption prediction. This structure contains the utility and the time frame.

Exception Definitions

12.2.15 IPublishable

Interface Identity

This interface is provided by every component that needs to receive messages from the event channel. These components are scattered throughout the complete system. This because every component that needs to receive data from the event channel needs such an interface.

Resources provided

- void publishMessage(EventMessage message)
  This method is used to receive data from the event channel.

Data Type Definitions

- EventMessage
  The message received from the event channel.

Exception Definitions

12.2.16 IReqCache

Interface Identity

Resources Provided

- void updateCache(DataTransferObject data)
  Method to store data in the cache.

Data Type Definitions

- DataTransferObject
  A data object that contains the information of a record in the Data storage.

Exception Definitions

12.2.17 IScheduler

Interface Identity

This interface is provided by the Scheduler for incoming alarm trames and the Scheduler for incoming measurement trames.

Resources provided

- void scheduleTrame(Trame incomingtrame)
  This method should be called to add a new trame to the scheduler.

Data Type Definitions

- Trame
  The trame that needs to be added to the scheduler for processing.

Exception Definitions

12.2.18 ISender

Interface Identity

This interface is provided by the Outgoing gateway. At the moment it is the only component providing this interface.

Resources Provided

- void sendSms(UserInfo userInfo, MessageContent content)
  This method will send a message using SMS with the given arguments.
- void sendGprs(UserInfo userInfo, MessageContent content)
  This method will send a message using gprs with the given arguments.
- void sendTcpIp(UserInfo userInfo, MessageContent content)
  This method will send a message using TCP/IP with the given arguments.

Data Type Definitions

- UserInfo
  Data type that contains information of a user.
- MessageContent
  The content of the message.

Exception Definitions

- IncompleteDataException This exception can be thrown by all three methods of the ISender. It is thrown when the UserInfo object does not contain sufficient data to send the message.

12.2.19 ISubscribable

Interface Identity

This interface is provided by the Event channel component. These components are scattered throughout the whole system. This is because several components need this channel to receive updates about the status of the system.

Resources provided

- void subscribeToTopic(EventDescription topic)
  This method can be used to subscribe to certain events that are sent through the event channel.
- void sendEvent(EventMessage event)
  This method allows components to send events through the event channel.

Data Type Definitions

- EventDescription
  The specific type of a message.
- EventMessage
  The message sent through the event channel.

Exception Definitions

12.2.20 ITrameSender

Interface Identity

This interface is provided by the Scheduler for outgoing trames.

Resources provided

- void sendTrame(TrameContent content)
  This method is used to add content to the queue that needs to be sent as a trame.

Data Type Definitions

- TrameContent
  The content that the trame must contain.

Exception Definitions

12.2.21 IUIS

Interface Identity

This interface is provided by the component that is responsible for the connection with the UIS.

Resources Provided

- UserInfo getCustomerInfo(UserInfo userInfo)
  This method will request all info that the UIS has of a customer. The method will pass a UserInfo object (that not necessarily needs to contain all information of a customer, only a predefined set of info which the UIS needs to identify the correct customer) and return a UserInfo object (which can contain more or different information of a customer).

Data Type Definitions

- UserInfo
  Data type that contains information of a user.

Exception Definitions

- IncompleteDataException
  This exception can be thrown by getCustomerInfo(UserInfo userInfo). It is thrown when the UserInfo object does not contain sufficient data to correctly identify the customer.

- CommunicationException
  This exception can be thrown if the component fails to establish a connection with the external source.

12.2.22 IUserInteraction

Interface Identity

This interface is provided by the Application controller component. The main use of this interface is interaction with users outside the ReMeS system.

Resources Provided

This interface has a wide variety of methods. Some of them are listed below.

- boolean login(AuthToken authToken)
  A method to login into the Application controller.
- void createBill(AuthToken authToken, UserInfo userInfo)
  A method to request a new bill creation.
- void giveFeedback(AuthToken authToken, Feedback feedback)
  A method the customer can use to give feedback to a detected anomy by ReMeS.
- void setAlarmRec(AuthToken authToken, UtilityType type, Recipient[] recipients)
  A method the customer can use to set the recipients for certain alarms.
- ConsumptionPrediction reqPrediction(AuthToken authToken, PredictionType type)
  A method to request a new consumption prediction.
- void createUser(AuthToken authToken, UserInfo userInfo)
  A method to create a new user in the system.
- void freqReconfig(AuthToken authToken, FreqProfile newFreq)
  A method to reconfigure the frequency that a remote measurement module sends trames with.

This list of methods is incomplete. The IUserInteraction interface contains all kinds of methods for a user to interact with the system.

Data Type Definitions

- AuthToken
  The specific piece of information that can authenticate a user.
- UserInfo
  Data type that contains information of a user.
- Feedback
  Data type that contains the feedback of a customer. This feedback can be positive or negative.
- UtilityType
  The type of utility (gas, water, power).
- Recipient
  Data type that contains information on who to contact and how to contact that person.
- ConsumptionPrediction
  A computed consumption prediction.

- PredictionType
  The type of the requested consumption prediction. This structure contains the utility and the timeframe.
- FreqProfile
  Data type that contains information on the frequency that a remote measurement module sends trames with.

Exception Definitions

- UnauthorizedException
  This exception can be thrown by all methods of this interface (except for the login method). This exception is thrown when the requested method cannot be executed by the user.

12.2.23 IUserNotifier

Interface Identity

This interface is only provided by the User notification component.

Resources provided

- void notifyUser(UserInfo userInfo, MessageContent notificiation)
  Use this method to add content to the queue that needs to be sent to a certain user.

Data Type Definitions

- UserInfo
  Data type that contains the information of a user.
- MessageContent
  The content of the message.

Exception Definitions