Instituto Superior Técnico - UL

TÉCNICO LISBOA

Enterprise Integration

IoT Project

Home Security and Automation

SafeHomeIoT
SECURITY & AUTOMATION MADE SIMPLE

Authors:	*IST ID:*
Diogo Moura	86976
Pedro Pereira	90766

19 of March of 2021

Contents

1 Introduction

This project aims to develop business processes and integrating them into an IoT company.

2 Company name and Logo

The proposed company name is "SafeHomeIoT". The proposed company logo is represented on figure 1.

SafeHomeIoT
SECURITY & AUTOMATION MADE SIMPLE

Figure 1: "SafeHomeIoT" logo

3 Services Provided

The company provides services in Home Security and Automation.
Our company area is the IoT. The services are provided through IoT devices and a central server, connected through a Wi-Fi/5G Internet connection.

4 Target Customers

Final consumers that rely on our channel and infrastructure to connect their IoT devices.
Companies that need a platform to connect their devices and reach the final consumer.

5 Value proposition

Our value proposition relies on two fundamental components, which can be used either by final consumers or companies:

1. Services: Up to Date Functional and valuable software for home automation that increases utility and security for customers.
2. Infrastructure: Highly reliable, secure, and easy to install channel/API to communicate with IoT devices over a Wi-Fi/5G Internet connection.

6 Business Drivers

There are two fundamental business drivers, each with different KPIs.

1. **Increasing customer satisfaction**
   This is related to improving customers' experience through the functionalities of the automation and security provided by our services.

   KPIs:

   - Average delay time between an order's scheduled date and the date it arrives and the average amount of products missing from an order.
     Measurement: Messages exchanged with supermarkets and customers who make orders allow to compute these statistics.
   - Average raise alarm precision (positive predicted value), this is, the fraction of alarms raised that represent real dangerous situations. Average raise alarm recall (true positive rate), this is, the the fraction of dangerous situations by the raise alarm. This is can be calculated because the user can report the undetected occurrences.
     Measurement: Messages exchanged with customers related to the real danger of the situations flagged by the alarms allow to compute these statistics.

2. **Increasing Business Efficiency**
   The service is provided directly through the internet available central server, which communicates with the IoT devices, by opposition to a home installed central, which allows to decrease the company costs.

   KPIs:

   - Number of dislocations to the users' house. Reduced number of dislocations indicates the central server solution is working well, as opposed to a home installed central, allowing the company to reduce costs with installment and technical support.
     Measurement: Messages exchanged with customers and technicians related with technical help allow to compute this statistic.

7 Operational Model

The company operational model is divided into the following components:

1. **Discover Service** Users discover the product through several channels: online marketing/physical stores/online stores where the IoT devices are sold. IoT devices documentation also describes the offered service and model.

2. **Select Service** Final Customers select the type of automation/security services on our website/app and choose the IoT devices on physical/online stores.

3. **Contract Service** Customers pay a monthly/annually fee for the service with online payments through our website/app.

4. **Consume Service** The service is consumed by final consumers through a Mobile/Web App/SMS that send the relevant information to the customer and provide interaction and authentication

8 Information Flow

The information flow is comprised of messages exchanged in the context of the Business Processes, described in the next section.

The main entities involved in the message exchanges are the Customers , Technicians, Supermarkets, wich contain orders and products, IoT devices and Services that contain Data Analyzers, associated to Subscriptions and IoT devices.

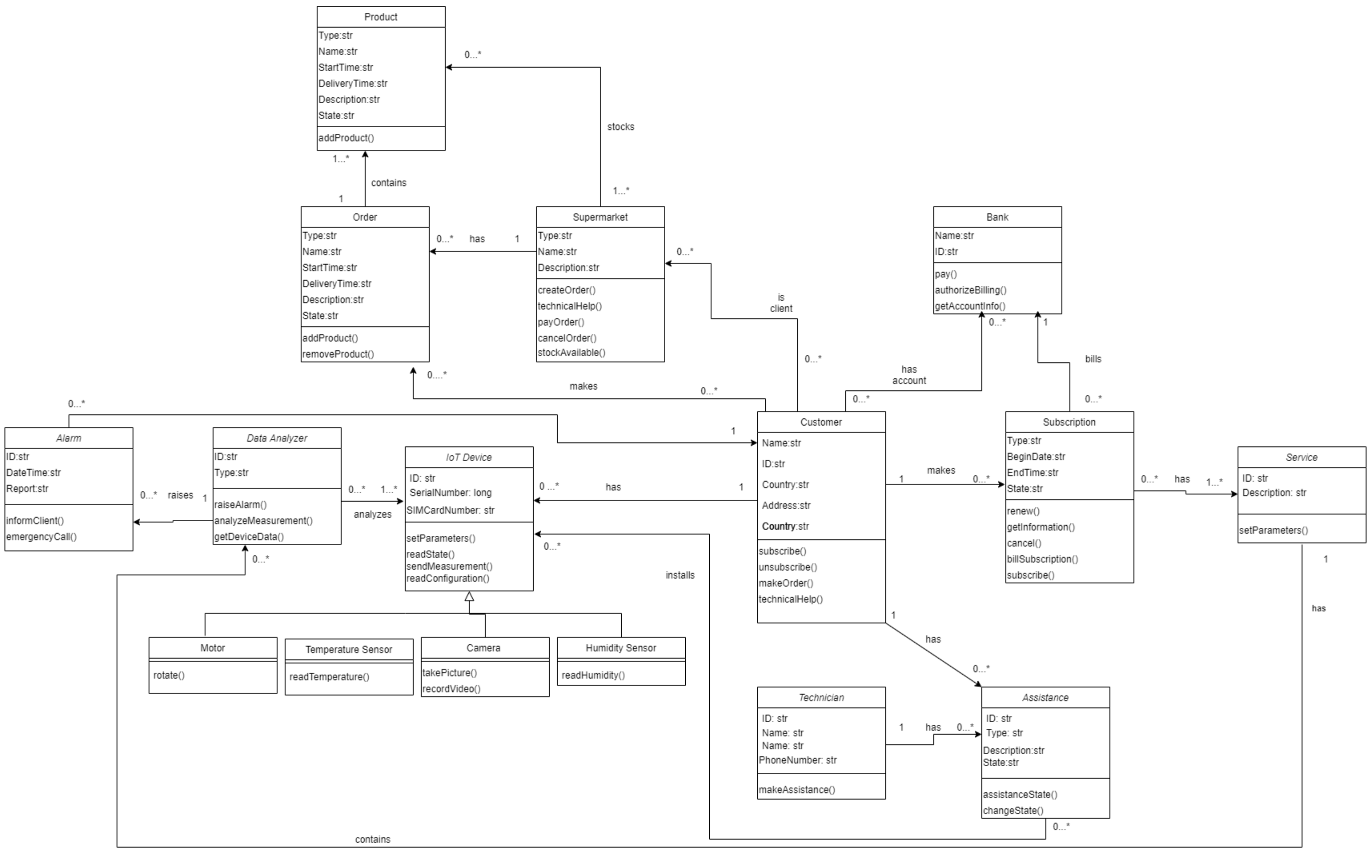

Figure 2: Information Flow

9 Business Processes

9.1 Subscribe to Service

The user accesses the web site and requests the available services (Catalog). The server retrieves them from a database and returns them to the user. The user then chooses the ones he is interested in. If the user is already logged in, he sends the server the services he chose. Otherwise, the user will have to fill a form with personal information, send it to the server, which then registers the new user. If the server waits more than 5 minutes to receive the chosen services from the user the process finishes. The server then initiates the payment process by contacting the bank services. The bank services the payment details which then, the server forwards to the user. If the user does not pay the process finishes. If he pays, he sends the data to server and the server forwards it to the bank services. The bank services sends the payment confirmation to the server and the server sends it to the user. Finally the server associates the services with the user on a database.

The business process diagram is represented on the Appendix, figure 4 and also on an external file named *subscribe_ to_ service.bpmn.*

9.2 Unsubscribe to Service

The user requests the server to unsubscribe to services. The server cancel the bank billing by connecting to its services and waits for a confirmation from the bank. Then dissociates the user from the services and confirms the unsubscription by sending him a message.

The business process diagram is represented on the Appendix, figure 5 and also on an external file named *unsubscribe_ to_ service.bpmn*

9.3 Add/Remove IoT device

The client intends to register a new IoT device in the network. So he makes a request to the server with the device information (e.g. serial number, type, brand, model). The server requests a SIM card activation to the telecommunication provider. It waits for the successful activation and informs the client. The server then registers the device and associates it with the customer. The server sends encryption keys for communicating in the network. The use then decides if he needs help. If he does, he sends an help request to the server, with the date the desired date for the installation. The server schedules a installation. A technician when the installation date arrives goes to the customer's house to perform the installation and marks it as complete. If the client does not need help he will perform the installation on his own and inform the server that the installation was successfully completed. If some reason during the installation the user has problems he will ask the server for help.

The business process diagram is represented on the Appendix, figure 6 and also on an external file named *add_ remove_ iot_ device.bpmn*

9.4 Change configuration

The client accesses the web app (Self-Service CRM) where he can: add/remove an IoT device from the network, change an Iot device or service configurations, read an IoT device config/state, control IoT device actions or leave the web app.

Requesting to add/remove a IoT device from the network will start the add_remove_iot_device process. This action updates the IoT device state on the server.

Requesting to change a device/automation configuration will result on the configuration parameters being updated on a database, will result on a message sent to the device, which will rewrite its configuration. After completing it, the IoT Device informs the server which inform the client that the parameter change was successful. This action updates the IoT device state on the server.

Requesting for an IoT to take an action will result in a message sent to the server which forwards that command to the IoT device. The device does the action updates its state and informs the server that the action has been taken, which then informs the client. This action updates the IoT device state on the server.

Reading the IoT Device will send a request to the server which forwards the request to the IoT device. The IoT device sends back the configuration and the server sends it to the user. This action updates the IoT device state on the server.

The business process diagram is represented on the Appendix, figure 7 and also on an external file named *change_ configuration.bpmn*

9.5 Raise Alarm

Sensors installed at the customer's house are constantly scanning the environment, whether it is measuring temperature or collecting images or measuring humidity or motion and heat signatures detection. These sensors are constantly sending the data they collect to a server. The servers logs the data from the sensors and analyzes it in an attempt to detect a potential occurrences. If the data doesn't display any evidence of occurrences, the process terminates. If the suspicion level is very high or confirmed, the server informs the client that a occurrence has been detected and calls the emergency services. The emergency services then go to the customer's house to see to the situation. If the suspicion level is medium or undetermined, the server asks the client to confirm if it is a real occurrence by sending him some information details. If the client does not detect any anomaly, the process terminates. If he detects, the server calls the emergency services. In the end, the occurrence is registered in a database table, even if it is a false negative (i.e. the server detected a possible occurrence, but it was not considered one by the client).

The business process diagram is represented on the Appendix, figure 8 and also on an external file named *raise_ alarm.bpmn*

9.6 House Inventory Management

The customer scans the products, adding or removing them to the stock. Every time a product is added or removed the sensor sends a message to a server which keeps track of each user's inventory. Whenever a product is removed the server check for any inventory shortages of that product. If there are any the it updates the customer's shopping list. When a product is added it only updates the user's inventory. Periodically some time before a certain day/hour of the week, defined by the customer the server makes an order to a shop/supermarket of the products in the shopping list. If the supermarket is unable to fulfill the delivery it informs the server, which informs the client. If it's able,it also informs the server which inform the client. The client will wait for the supermarket to deliver the products if the products aren't there on time, the client will inform the server the delay. The server then asks the supermarket for the delivery state and the server forwards it to the client. The shop eventually delivers the products to the customer's house, which then are stored in the shelves/fridge.

The business process diagram is on the Appendix, figure 9 and also on an external file named *house_ inventory_ management.bpmn*

10 Regulatory Compliance

There are already some standards for IoT communication, namely IEEE standards such as:

1. Infrastructure (ex: 6LowPAN, IPv4/IPv6, RPL)
2. Identification (ex: EPC, uCode, IPv6, URIs)
3. Comms / Transport (ex: Wifi, Bluetooth, LPWAN)
4. Discovery (ex: Physical Web, mDNS, DNS-SD)
5. Data Protocols (ex: MQTT, CoAP, AMQP, Websocket, Node)
6. Device Management (ex: TR-069, OMA-DM)

7. Semantic (ex: JSON-LD, Web Thing Model)

8. Multi-layer Frameworks (ex: Alljoyn, IoTivity, Weave, Homekit)

Our plataform will rely on some of these standards, namely IPv6/IPv4, WIFI, 5G, JSON.

There is generally a lack of IoT regulation. However, in some places, there is some regulation in place related to the security of the IoT communication channel. Namely devices must be patchable, rely on industry standard protocols, and be built without hard-coded passwords and known security vulnerabilities (https://innovationatwork.ieee.org/should-government-regulate-iot/).

Our platform aims at complying with these regulations, through providing a secure communication channel for IoT devices.

11 IoTaaS architecture

The system architecture has the following communication patterns and entities:

1. Provisioning activates/suspends devices and informs the 1st mediation layer that new Topics for new Users need to be created.

2. IoT Devices send messages/events to a single topic running on Kafka Brokers named EventsTopic.

3. The 1st Mediation Layer reads the IoT devices messages from the EventsTopic and writes to a topic for each Client. It also reads from a specific Topic messages sent by Provisioning, which indicate if a new topic for a client needs to be created and creates the topic for the client.

4. The second mediation Layer inserts the messages into a database after reading from the client's topic and makes the data available to an upper layer through a function named getNextEvent().

On figure 3 the system entities relationships are represented.

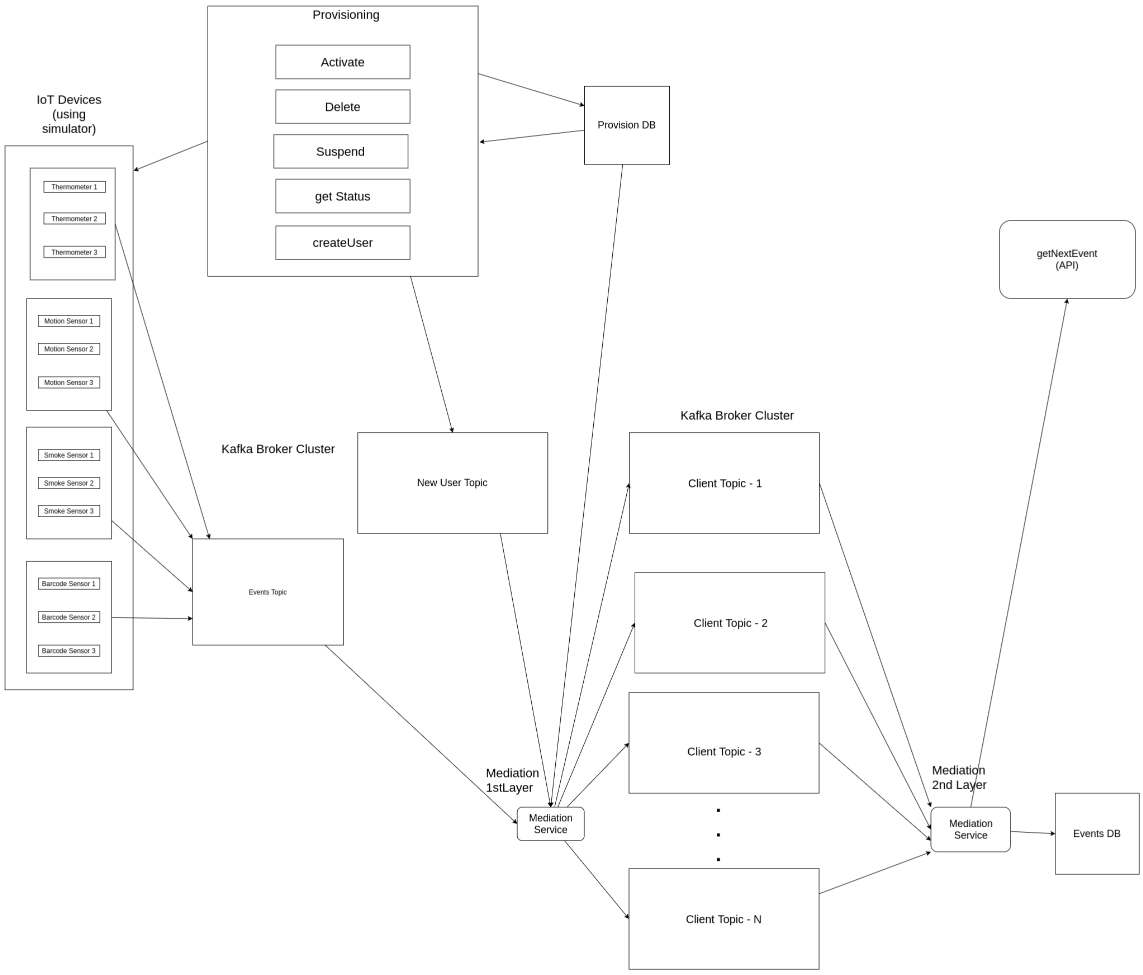

Figure 3: Architecture of the system

12 Radio Network

A database is created in Provisioning with two tables. One table contains the users that are active and other the users that are suspended.

```
CREATE DATABASE HLR;
CREATE TABLE  HLR.activeSubscriber (SIMCARD VARCHAR(22), MSISDN VARCHAR(15)
                , userID INT);
CREATE TABLE HLR.suspendedSubscriber (SIMCARD VARCHAR(22), MSISDN VARCHAR(15)
                , suspendedSubscriber INT);
```

A program takes into account the devices registered in the database to simulate the radio network activity.

The type of messages simulated are:

1. temperatureMessage:{ "ID":10, "deviceID":3, measurement:28.3, type:"temperature", ts:"2021-01-19 03:14:07"}

2. imageMessage:{ "USERID":10, "deviceID":3, description:"cat", type:"image", ts:"2021-01-19 03:14:07"}

3. videoMessage:{ "USERID":10, "deviceID":3, description:"cat moving", type:"video", ts:"2021-01-19 03:14:07"}
4. smokeMessage:{ "USERID":10, "deviceID":3, measurement:0.7, type:"smoke", ts:"2021-01-19 03:14:07"}
5. motionMessage:{ "USERID":10, "deviceID":3, measurement:1 , type:"smoke", ts:"2021-01-19 03:14:07"}

These messages simulate the several type of IoT devices supported.
These messages are then exchanged through the system layers.

13 Provisioning

Devices are simulated in the radio network for the devices with the SIMCARD and MSISDN defined in the activeSubscriber table. There are four operations defined in provisioning:

1. activate(SIMCARD,MSISDN,userID)
   The device identified by SIMCARD and MSISDN is added to the activeSusbcriber table together with the userID and (if needed) removed from the suspendedSubscriber table.
2. suspend(SIMCARD,MSISDN,userID)
   The device identified by SIMCARD and MSISDN is removed from the activeSubscriber table and added to the suspendedSubscriber table together with the userID of the owner.
3. delete(SIMCARD,MSISDN)
   The device identified by SIMCARD and MSISDN is removed both from the activeSubscriber and suspendedSubscribe tables.
4. getStatus(SIMCARD,MSISDN)
   The current status of the device identified by SIMCARD and MSISDN is returned from the database, whether it is active or suspended.

14 Mediation

The Mediation has two layers:

1. 1st Layer
   It consumes the messages from the EventsTopic, that is served by the simulator. Then a new topic is chosen to which the messages are produced, based on the client that is the owner of the device, as there is a topic for each different client. For example, for a message with the following signature "ID":10, "deviceID":3, measurement:1 , type:"smoke", ts:"2021-01-19 03:14:07", the message would be consumed and then the same message would be produced to a new topic in the kafka broker with the name usertopic-10.
2. 2nd Layer
   It consumes the messages from the clients topics and saves them to a database. It also makes the data available to an API upper layer, which will be developed in the

next sprint. The API should call the method getNextEvent() from the 2nd Mediation Layer, which is already developed. The database to which the messages are saved has its schema defined by the following code:

```
###########SENSORS#################
CREATE DATABASE IF NOT EXISTS SafeHomeIoTEvents;
USE SafeHomeIoTEvents;

DROP TABLE IF EXISTS temperatureMessage;
CREATE TABLE temperatureMessage (
        ID INT AUTO_INCREMENT,
        deviceID INT,
        measurement FLOAT,
        type VARCHAR(30),
        ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        userID INT,
        PRIMARY KEY(ID)
);

DROP TABLE IF EXISTS imageMessage;
CREATE TABLE imageMessage (
        ID INT AUTO_INCREMENT,
        deviceID INT,
        description VARCHAR(30),
        type VARCHAR(30),
        ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        userID INT,
        PRIMARY KEY(ID)
);

DROP TABLE IF EXISTS videoMessage;
CREATE TABLE videoMessage (
        ID INT AUTO_INCREMENT,
        deviceID INT,
        description VARCHAR(30),
        type VARCHAR(30),
        ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        userID INT,
        PRIMARY KEY(ID)
);

DROP TABLE IF EXISTS smokeMessage;
CREATE TABLE smokeMessage (
        ID INT AUTO_INCREMENT,
        deviceID INT,
        measurement FLOAT,
        type VARCHAR(30),
        ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        userID INT,
        PRIMARY KEY(ID)
);

DROP TABLE IF EXISTS motionMessage;
CREATE TABLE motionMessage (
        ID INT AUTO_INCREMENT,
        deviceID INT,
        description VARCHAR(30),
        type VARCHAR(30),
        ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        userID INT,
        PRIMARY KEY(ID)
);
```

15 Kafka Cluster

The Kafka Cluster is used to receive the messages from the devices by the mediation layers. There is one topic for receiving the events from the Radio Network, one for receiving the events that indicate that a new user was created and that a new topic for that user needs to be created and N topics, 1 for each of a total of N clients.

Here is the calculation used to optimize the number of partitions for a Kafka implementation.

$$\#Partitions = \left\lceil \frac{TopicThroughput}{PartitionSpeed} \right\rceil \tag{1}$$

Assuming each client generates a peak traffic of 5 MB/s, the traffic required for creating new users is 1 MB/s and assuming conservatively that the Partition Speed is 10MB/s:

- the number of partitions for the Events topic should be $\left\lceil \frac{\#Clients*ClientTraffic}{PartitionSpeed} \right\rceil = \left\lceil \frac{\#Clients}{2} \right\rceil$.
- the number of partitions for the Events topic should be $\left\lceil \frac{ClientTraffic}{PartitionSpeed} \right\rceil = 1$
- the number of partitions for the topic that indicate that a new user was created should be $\left\lceil \frac{TopicThroughput}{PartitionSpeed} \right\rceil = 1$

The number of brokers and clusters should also be sized according to the number of clients. A general good rule is that a broker should not hold more than 2000 to 4000 partitions (across all topics of that broker) and a Kafka cluster should have a maximum of 20,000 partitions across all brokers.

According to these rules:

$$\#Partitions = \left\lceil \frac{\#Partitions}{4000} \right\rceil \approx \left\lceil \frac{\#Clients/2}{4000} \right\rceil = \left\lceil \frac{\#Clients}{8000} \right\rceil \tag{2}$$

$$\#Clusters = \left\lceil \frac{\#Partitions}{20000} \right\rceil \approx \left\lceil \frac{\#Clients}{40000} \right\rceil \tag{3}$$

These are theoretical calculations that are used for illustration purposes or if the number of brokers/clusters is defined statically this can be used as a rough estimate. The reference for these computations can be found under Dattell's Kafka, Pulsar, and Elastic solutions blog link.

Ideally a solution is to use AWS Auto-Scaling with a trigger based on the Kafka amount of traffic/requests or to use AWS Kafka MSK API to increase/decrease the number of brokers/clusters dynamically, instead of having a static number of brokers/clusters.

16 Implementation

The implementation can be found under the SafeHomeIoT project root folder. The project is constituted by different modules which should be run on different platforms. The project should be run on different virtual machines/serverless functions (aws lambda). Each jar file corresponding to each module is deployed on each of these different platforms. There should also be Kafka broker cluster running on AWS EC2. The jar files of the modules that run on AWS Lambda are deployed directly to the cloud. The jar files of the modules that run on virtual machines should run with the following command, from the module root folder:

```
# mvn exec:java <arguments>
```

- provisioning - AWS Lambda
  Deployed directly to AWS Lambda and run with no arguments

- simulator - Ubuntu Virtual Machine (or any machine) with kafka producer that produces all the messages from the IoT devices to a Kafka Topic named EventsTopic.
  Run the command:

  ```
  #  mvn clean compile exec:java -Dexec.args="-telecommunications-provider-name SafeHomeTelco
  -broker-list {KAFKA_BROKER_IP}:9092 -topic safehomeiot-events -hlr-server {DB_ENDPOINT}
  -hlr-database HLR -hlr-username {DB_USER} -hlr-password {DB_PASSWORD}
  -hlr-table activeSubscriber
  ```

- eventmediationpre - 1st Layer Mediation
  Ubuntu Virtual machine with a kafka Consumer and a Producer that consume messages from the EventsTopic provisioned by the simulator and write to several different topics, based on the client that each IoT device is associated to the events are written to a different topic for each client. Run the Command:

  ```
  # mvn exec:java -Dexec.args="-kafkaip {KAFKA_BROKER_IP}:9092 -topics safehomeiot-events
  ```

- events
  This module shouldn't be called directly on any machine. It just contains event class definitions that are used in the eventmediation module.

- eventmediation
  2nd Layer Mediation
  Ubuntu Virtual machine with a kafka Consumer that reads messages from the several topics. There is one topic for each client, so events are consumed from several topics each for a different client. The events are written to a database and are also available to an upper layer to be consumed.

Run the command:

```
# mvn clean compile exec:java -Dexec.args="-kafkaip {KAFKA_BROKER_IP}:9092"
```

- eventserververvice
  AWS lambda function that implements getNextEvent() by accessing the Event-Database. It receives JSON as input with userId, typeEvent and lastReceivedId. It returns the events of the given type, belonging to the given user whose id is greater than lastReceivedId.

17 Testing

To test the system, the deployment configuration described previously in Implementation was used.

The Kafka Broker Cluster configuration used was the following:

- Number of Topics: Number of Clients+2 (EventsTopic+UserTopicCreationTopic)
- Number of Partitions per Topic: 1
- Replication Factor:1
- Number of Brokers:1
- Number of Clusters:1

The following IoT devices were added to the simulator through the Provisioning database:

```
INSERT INTO activeSubscriber (SIMCARD , MSISDN)
                VALUES (173864374,911234567, "temperature");
INSERT INTO activeSubscriber (SIMCARD , MSISDN)
                        VALUES (273864374,911234568, "motion");
INSERT INTO activeSubscriber (SIMCARD , MSISDN)
                                VALUES (373864374,911234569, "smoke");
INSERT INTO activeSubscriber (SIMCARD , MSISDN)
                                VALUES (373864574,911234570, "video");
INSERT INTO activeSubscriber (SIMCARD , MSISDN)
                VALUES (373964574,911234571, "image");
```

.

The Clients' topics were dynamically added to the Kafka Cluster automatically by the 1st Mediation Layer, signaled by the Provisioning Layer.

The 2nd Mediation Layer inserted the Users' messages in a database and made the messages available to an Upper Layer through the function getNextEvent().

The system behaved as expected and all the functionalities run correctly.

More brokers/clusters should be used to test the scalability and reliability of the architecture. A solution could also be to use AWS Auto-Scaling to increase the number of clusters dinamically or to use the AWS Kafka MSK API.

Appendix

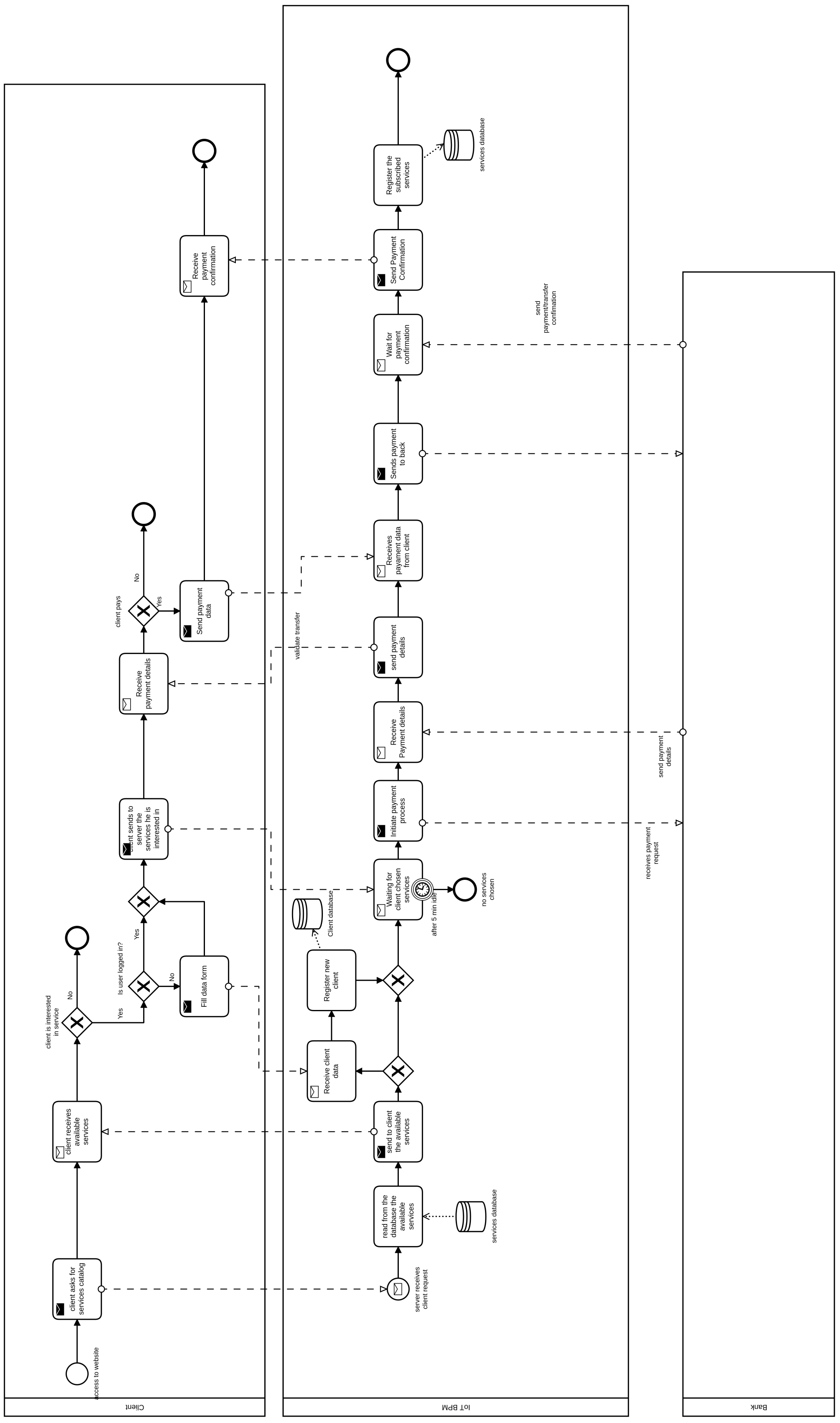
Client
IoT BPM
Bank
access to website
client asks for services catalog
client receives available services
client is interested in service
No
Yes
Is user logged in?
Yes
No
Fill data form
client sends to server the services he is interested in
Receive payment details
client pays
No
Yes
Send payment data
Receive payment confirmation
server receives client request
read from the database the available services
services database
send to client the available services
Receive client data
Register new client
Client database
Waiting for client chosen services
after 5 min idle
no services chosen
Initiate payment process
Receive Payment details
send payment details
validate transfer
Receives payament data from client
Sends payment to back
Wait for payment confirmation
Send Payment Confirmation
Register the subscribed services
services database
receives payment request
send payment details
send payment/transfer confirmation

Figure 4: Subscribe to Service

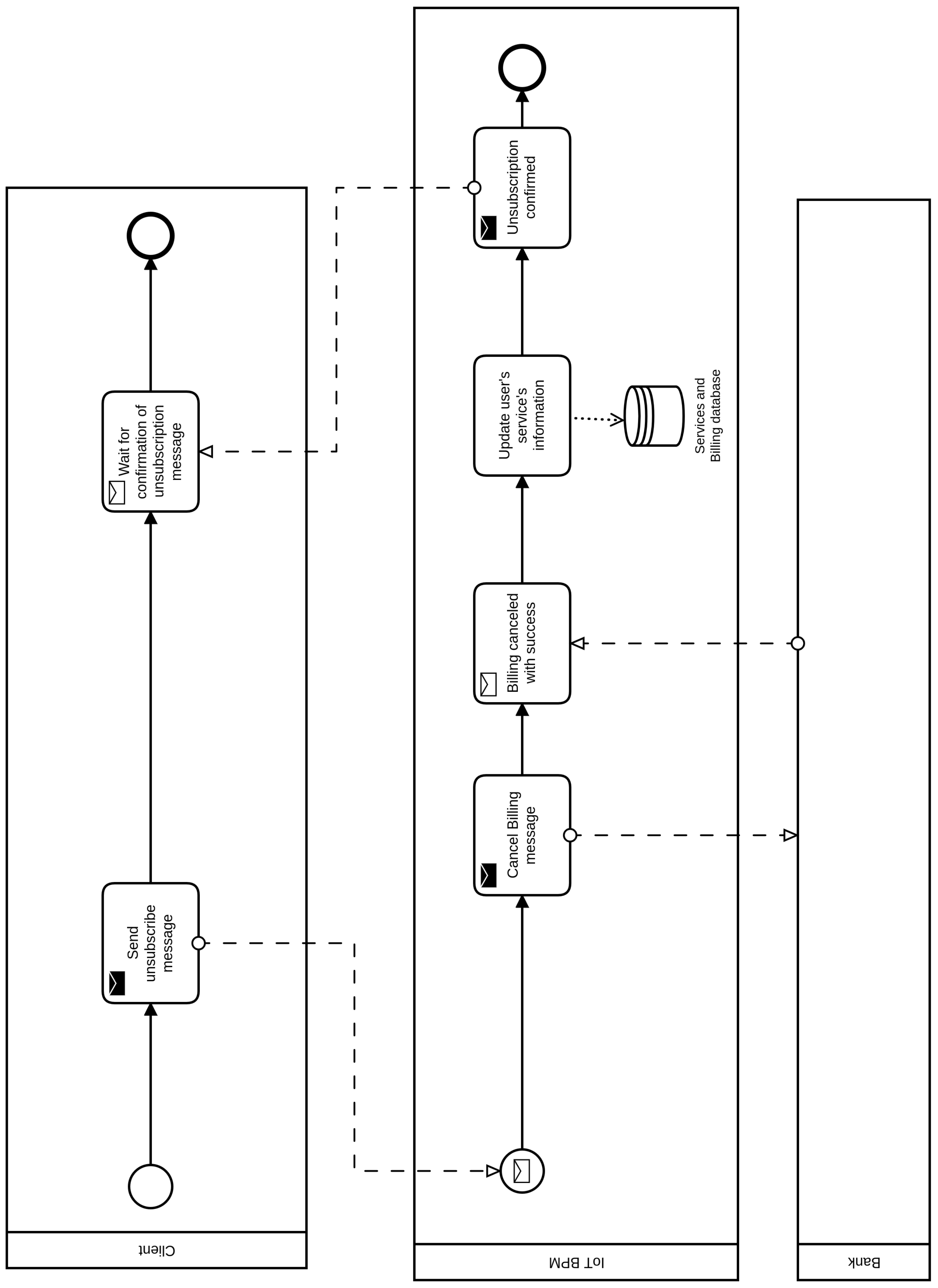
Client
Send unsubscribe message
Wait for confirmation of unsubscription message
IoT BPM
Cancel Billing message
Billing canceled with success
Update user's service's information
Unsubscription confirmed
Services and Billing database
Bank

Figure 5: Unsubscribe to Service

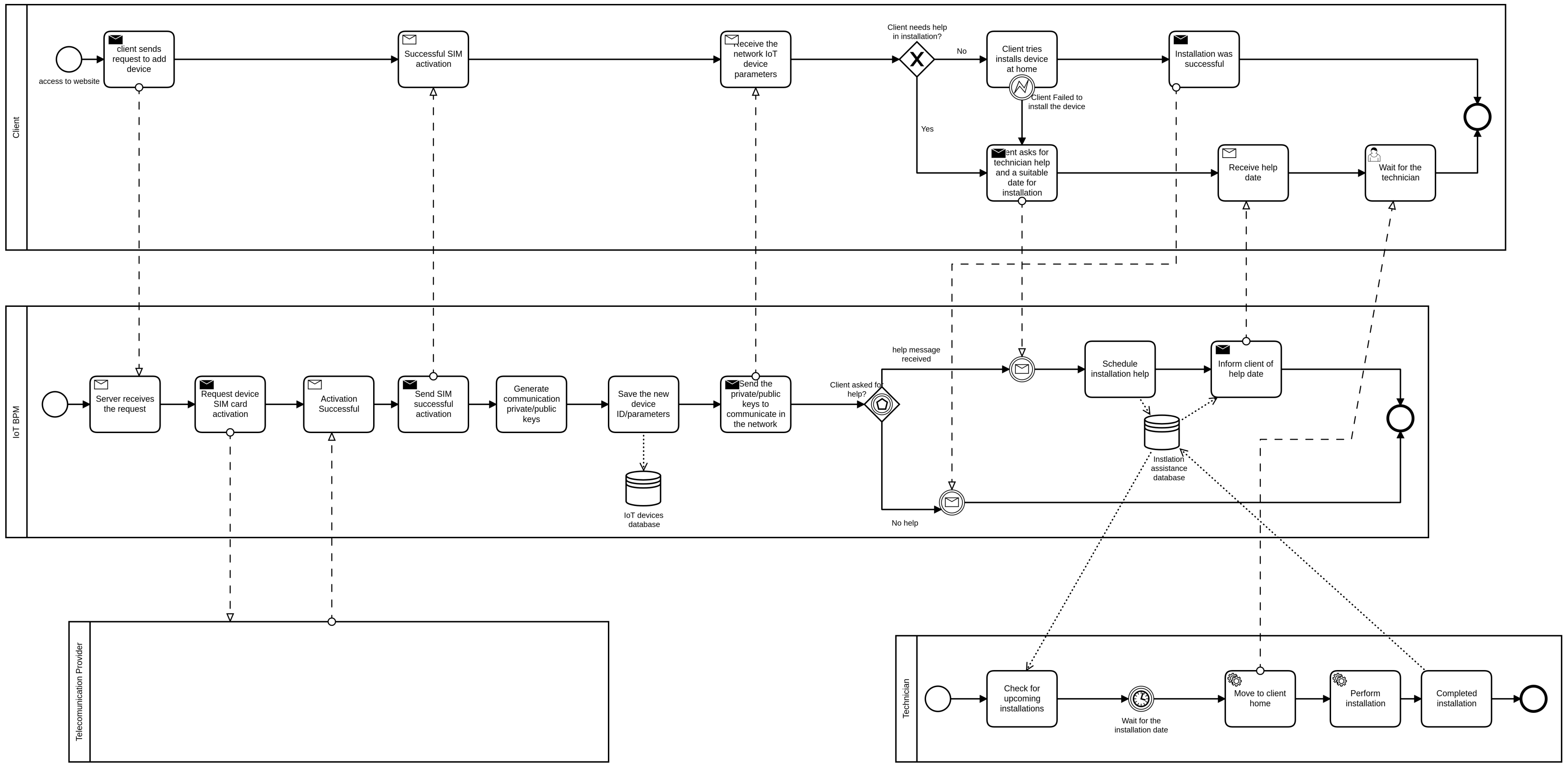
Client
access to website
client sends request to add device
Successful SIM activation
Receive the network IoT device parameters
Client needs help in installation?
No
Yes
Client tries installs device at home
Client Failed to install the device
ent asks for technician help and a suitable date for installation
Installation was successful
Receive help date
Wait for the technician
IoT BPM
Server receives the request
Request device SIM card activation
Activation Successful
Send SIM successful activation
Generate communication private/public keys
Save the new device ID/parameters
IoT devices database
Send the private/public keys to communicate in the network
Client asked fo help?
help message received
No help
Schedule installation help
Inform client of help date
Instlation assistance database
Telecomunication Provider
Technician
Check for upcoming installations
Wait for the installation date
Move to client home
Perform installation
Completed installation

Figure 6: Add/Remove IoT device

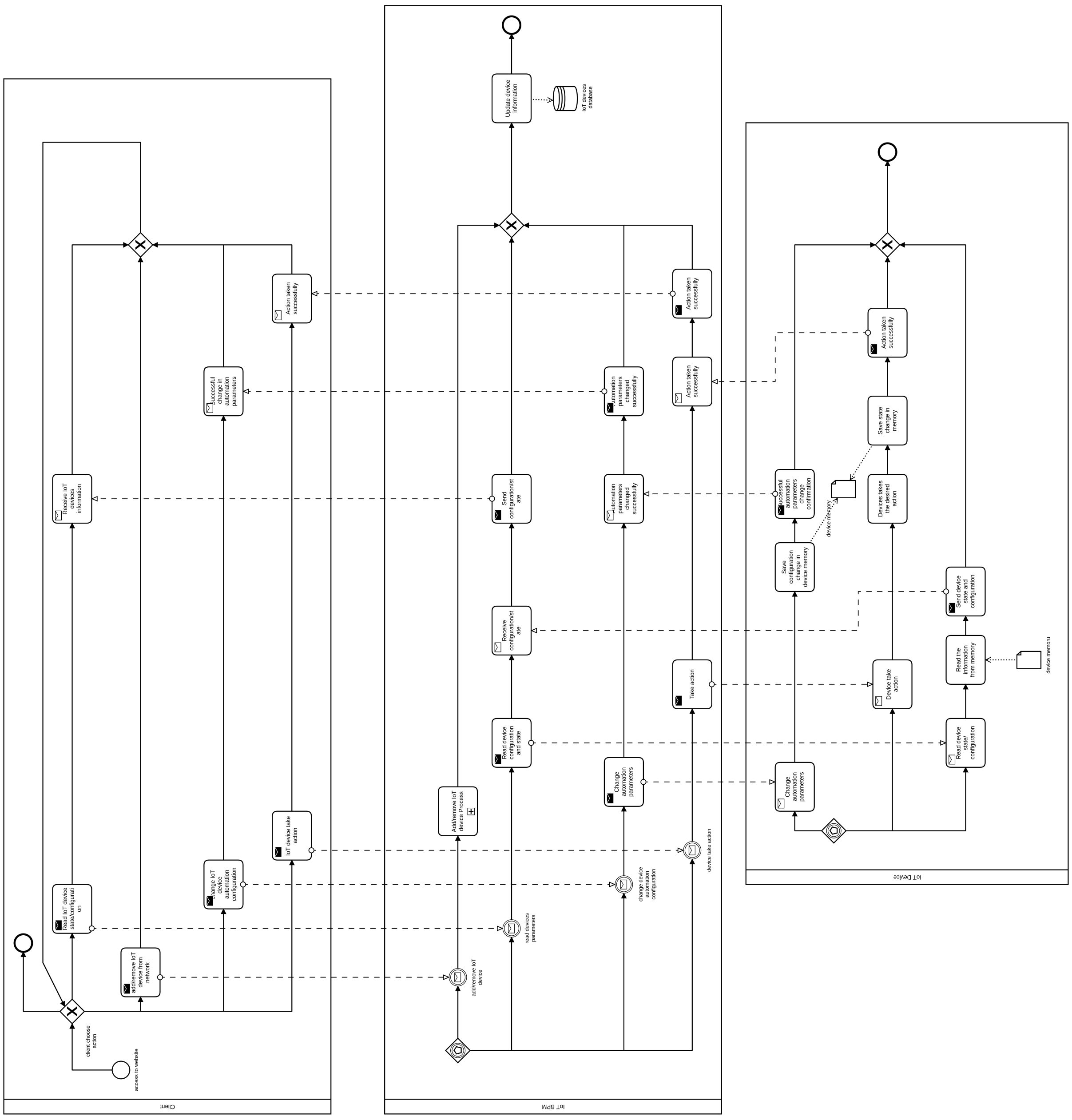
Client
access to website
client choose action
add/remove IoT device from network
Read IoT device state/configuration
Change IoT device automation configuration
IoT device take action
Receive IoT devices information
successful change in automation parameters
Action taken successfully
IoT BPM
add/remove IoT device
read devices parameters
change device automation configuration
device take action
Add/remove IoT device Process
Read device configuration and state
Receive configuration/state
Send configuration/state
Change automation parameters
automation parameters changed successfully
Take action
Action taken successfully
Update device information
IoT devices database
IoT Device
Change automation parameters
Save configuration change in device memory
successful automation parameters change confirmation
device memory
Device take action
Devices takes the desired action
Save state change in memory
Read device state/ configuration
Read the information from memory
Send device state and configuration

Figure 7: Change configuration

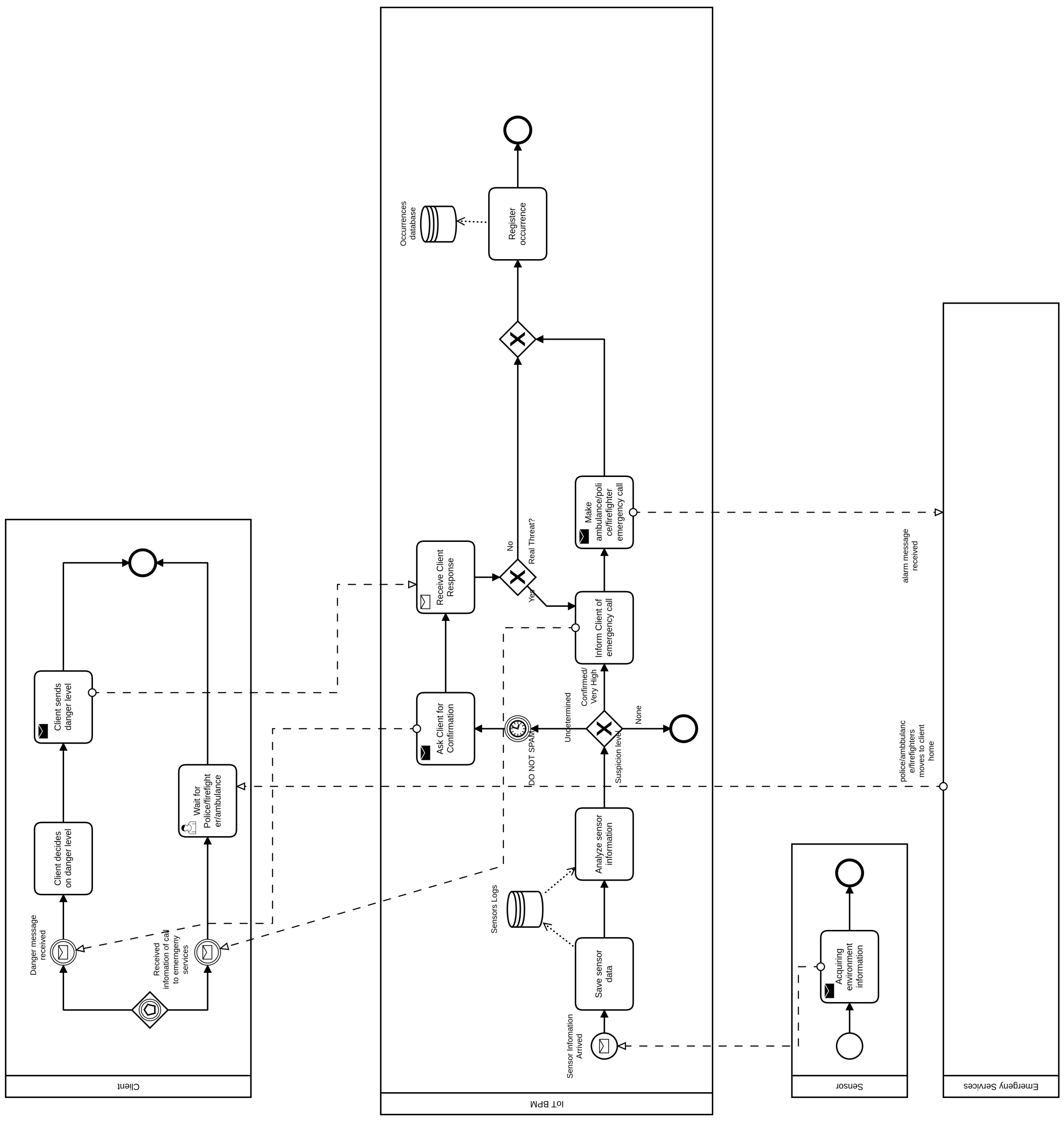
Client
Danger message received
Client decides on danger level
Client sends danger level
Received infomation of call to emergeny services
Wait for Police/firefighter/ambulance
IoT BPM
Sensor Infomation Arrived
Save sensor data
Sensors Logs
Analyze sensor information
Suspicion level
None
Undetermined
DO NOT SPAM
Confirmed/ Very High
Ask Client for Confirmation
Receive Client Response
Real Threat?
No
Yes
Inform Client of emergency call
Make ambulance/police/firefighter emergency call
Register occurrence
Occurrences database
Sensor
Acquiring environment information
Emergeny Services
alarm message received
police/ambulance/firefighters moves to client home

Figure 8: Raise alarm

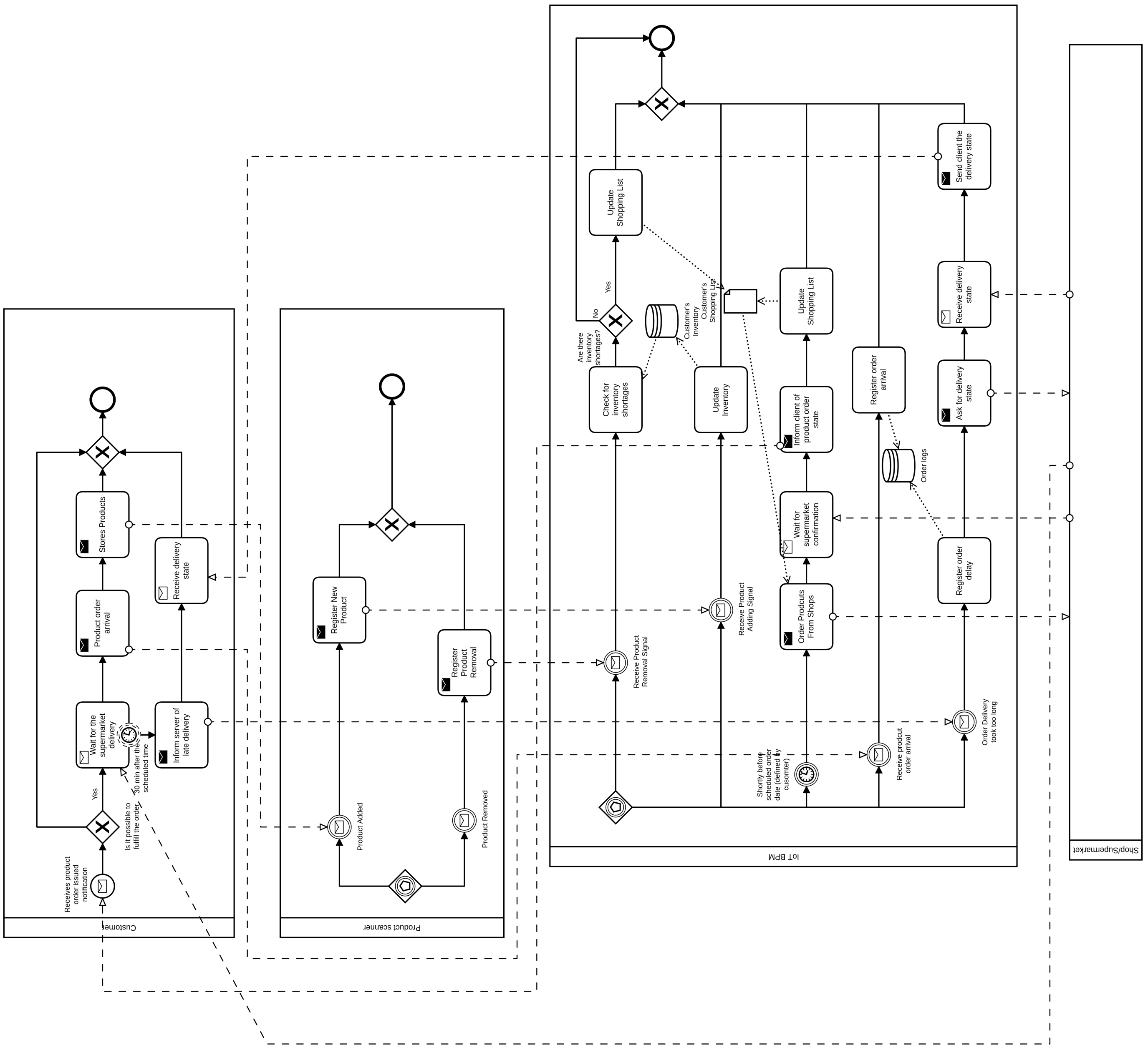
Customer
Receives product order issued notification
Is it possible to fulfill the order
Yes
Wait for the supermarket delivery
30 min after the scheduled time
Product order arrival
Stores Products
Inform server of late delivery
Receive delivery state
Product scanner
Product Added
Product Removed
Register New Product
Register Product Removal
IoT BPM
Receive Product Removal Signal
Receive Product Adding Signal
Check for inventory shortages
Are there inventory shortages?
No
Yes
Update Shopping List
Update Inventory
Customer's Inventory
Customer's Shopping List
Shortly before scheduled order date (defined by cusomter)
Order Prodcuts From Shops
Wait for supermarket confirmation
Inform client of product order state
Update Shopping List
Receive prodcut order arrival
Register order arrival
Order logs
Order Delivery took too long
Register order delay
Ask for delivery state
Receive delivery state
Send client the delivery state
Shop/Supermarket

Figure 9: House inventory management