A Model Driven Engineering through Architecture Description in e-Health Systems

Mohammad Faidi[1]

[1]An-Najah National University
[1]Faculty of Engineering and Information Technology
[1] departments of IT-CS

Absrtact

The health care industry is in a state of extreme despair and the services provided by health care are more expensive than ever. As the world's population grows, chronic diseases have also increased markedly. The technologies and devices that support Internet of Things have developed rapidly and their solutions are currently being applied on various domains of our life where the Internet of Things has made monitoring in the health care system possible and easy which helps maintain patient safety and helps doctors to provide the best care for patient and this saves time, effort and costs. This paper focuses on the healthcare domain. First of all, this work illustrates the CAPS modeling languages used to describe the software architecture which we customize a specific language metamodel of the SAML language, we can formalize the structure and constructs of the SAML languages for modeling any system related to the healthcare domain. In addition, to complete a process of modeling we gathered all the sensors used in this domain. As a simple case study of Healthcare based on Caps Framework, we established four architecture design as follow: Smart Hospital Gown Model, Smart Eyeglasses Model, Diabetes Model and Diabetes Mellitus Mobile application Model. Also, we build a specific simulator for Diabetes case study. It aims to evaluate how many doses of insulin patient needs. Moreover, design Mobile Application Diabetes mellitus type 1 patients have to help to take insulin injection 3 times a day with each meal(Bolus), and one or two injection long insulin acting(Basel). In our work, we used the CupCarbon for being able to evaluate the energy consumption of SAML Diabetes mellitus Mobile Application.

Keywords- **CAPS MDE Meta-Model Architecture-Description SAML**

1 Introduction

Elements of the Internet are becoming more and more ubiquitous. Internet elements (such as definition, sensing, communication, installation, etc.) are doing different tasks to build a continuous monitoring system using Internet elements. These elements must deal with each other and adapt themselves to the changes surrounding them and the existing environment in order to carry out the tasks assigned to it properly. In our work, we have modified the CAPS [3] metamodel, which is an architecture-driven modeling framework for the development of Situational Aware Cyber-Physical Systems.

1) **Overview:** The software architecture view allows designers to dene the software architecture of the application through the SAML modeling language. Software components exchange messages through message ports. Each component can declare a set of application data manipulated by actions dened in the behavior of the component. The behavior of each component is represented by a list of events, conditions, and actions. Modes can be dened as well.
2) **Meta-Model:** By dening the SAML underlying metamodel, we can formalize the structure and constructs of the SAML language. Figure 1 and Figure 2 show the parts of the SAML metamodel related to its structural and behavioral concepts, respectively. The software architecture of CAPS is dened as a collection of software components and connections. A component is a unit of computation with internal state and well-dened interface . The internal state of a component is denoted by the current behavioral mode of application data and its values. An application data can be shown as a local variable declared in the component scope; application data are deployed by actions, events, and conditions dened in the component behavior, such as StoreData, ReceiveMessage, etc

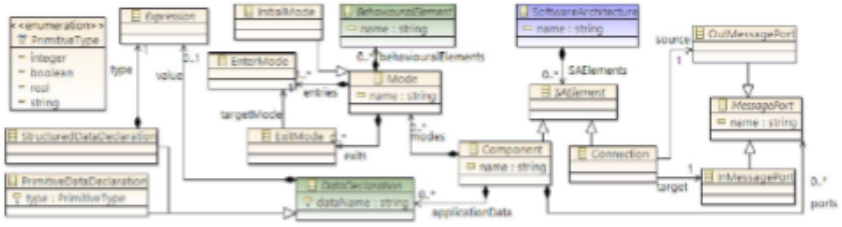

Figure 1: SAML Metamodel: structural concepts (external metaclasses in green)

A **mode** indicates a specific status of the component.Examples of modes can be, energy saving mode, sleeping mode, etc. The component can only have one active mode at a time. Mode transitions occur by passing from an action called **ExitMode** to a distinct type of event called **EnterMode**. Thus, entry and exit modes can

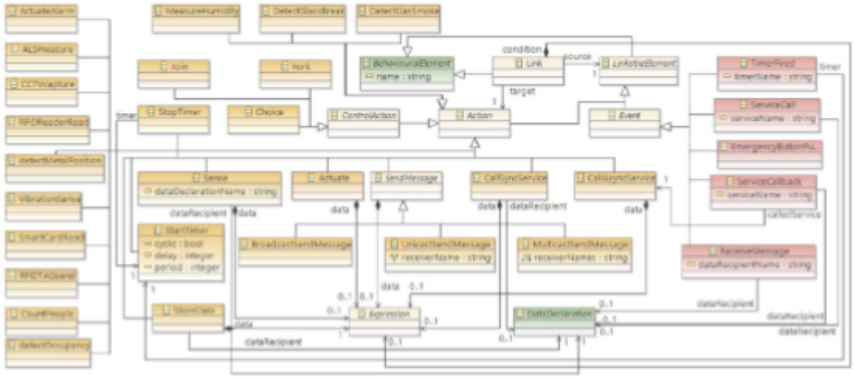

Figure 2: SAML Metamodel: behavioral concepts (actions in orange, events in red

be associated with actions and events, that formalize among modes a continuous behavioral flow. Every single mode can have a set of **behavioral elements** denoted by actions, conditions,and events that all together depict the control flow within the component.
Components can interact by passing messages through **message ports** . For receiving incoming messages input message ports are used while output message ports are used for sending outgoing messages. Actual communication methods of a message (i.e., broadcast, multicast or unicast) are shown in Figure 2. In this context, a **connection** represents unidirectional communication channel between two message ports of two different components.
An **action** is a special kind of behavioral element which represent an atomic task that can be performed by the component. The action can be executed when it is triggered by an event or when a previous action in the behavioral flow has been achieved. An action can be for example, get data from a specific sensor (e.g, occupancy sensor),start or stop of a timer, send a message from a specific port, and so on. Precise types of actions can be followed :the fork action splits the control flow into multiple parallel executions, the join action merges (previously split) control flows, and the choice action specifies a branch in the control flow after which one and only one outgoing control flow will be executed.
An **event** is triggered in response to either some internal appliance of the component (e.g., a timer fired) or an external stimulus of the component (e.g., the message reception on a input message port).

Examples of events: entering a specific mode, receiving a message at a given port, an activation of a timer, the receiving of a call from an external service, etc.
Finally, connections between events and actions that rep-resent the control flow among them are called links.An architect can benefit from these links in deciding the order in which actions can be performed and the actions that must be executed when an event is triggered.
As appropriate with the domain of our work which is a health care domain where the software architecture view allows designers to define the software architecture application through the SAML modeling language. And we added all the devices and sensors that could be needed by anyone who wants to build a model for the domain of health care and then we have built three models case study by using SAML.

THE SOFTWARE ARCHITECTURE VIEW MODELING LANGUAGE (SAML).

The first model for the Smart Hospital Gown. Although the great development in the domain of Internet of things, but still the hospital rooms for patients using devices and sensors attached to terminal devices to monitor the vital signs of the patient as temperature, which is measured by using a thermometer by inserting it into the mouth of the patient and also be monitored some other signs of life for the patient only visually or not monitored at all like measuring the rate of sweating in the patient and convulsions and chills, even biometric signs, which are monitored as a heart rate monitor and the rate of breath is measured and taken readings at specific periods of time and because this control is done by the use of some wired devices so this restricts the movement of the patient to move from room to room or even go to the bathroom, and his movement is limited only to the hospital bed where every time the patient wants to move from one side to another requires the separation of the screen wiring associated with the body of the patient and this is a problem in itself, which requires finding a solution, so we have done a suitable model for Smart Hospital Gown. This allows the patient to wear the gown easily instead of the normal dress that the patient is currently wearing in hospitals. This Smart Hospital

Gown is used to measure the patient's vital signs such as temperature, sweating rate and vibration for early detection of cold instead of a visual examination and also monitor the heartbeat and oxygen rate of the patient.

The second model for Smart Eyeglasses. Where the Internet of Things is used in the field of ophthalmology, we have done a model designed for probable dangerous pressure changes in human eyes. It works early detection of optic nerve disorder if it occurs within the eye where it can be inside the eye lack of the amount of blood supply to the inside of the eye, leading to disturbances in vision which can lead to blindness. The eyeglasses monitor eye pressure and specify that if there is a bleeding inside the eye or not.

The third model for diabetes. Diabetes is one of common disease in the world which may to death when there is no suitable health care. When you lose to control the disease, this effects on the life of the patients. The sugar level is helpful for the patient to take suitable doses in the actual time which makes patient in good situation. Internet of things can help monitor blood glucose in diabetic patients, helping them maintain blood glucose level within their normal range by helping them balance the intake of food, exercise and take insulin doses and thus reduce the incidence of complications to diabetic patients, there are many equipment and devices available which contribute to monitoring blood sugar and thus help diabetics. According to the World Health Organization's report, the number of people with diabetes exceeded 422 million people and in 2012 more than 1.2 million people died of diabetes, and classified by the World Health Organization diabetes as one of the ten reasons for death where it has a great and dangerous impact on the person infected with it and on the community as well. Unfortunately, there is no permanent treatment for diabetes. However, there is only one solution to this problem is to measure the level of sugar in the blood continuously and take the appropriate doses of insulin. ,Moreover, We built a simulator that relies on reading glycemic measurements for a diabetic patient and based on them determine how much insulin dose the patient should take to lower the high blood sugar.

The four model for Smart Diabetes mellitus Mobile Ap-

plication. Type 1 diabetic patients have to take insulin injection 3 times a day with each meal, and one to two injection long insulin. The patient must calculate the amount of insulin and carb in each meal.

2 BACKGROUND AND CASE STUDY

Smart Hospital Gown Case Study. We will simply represent an example of the Smart Hospital Gown case study. A simple scenario describes monitoring the patient's vital signs in the hospital, namely oxygen rate, heart rate, vibration rate, sweating and temperature through patient wear a Smart Hospital Gown. We will show its SAML model. Figure 3 shows the SAML model . It is important to note that this figure is actually a screen-shot of our Model. From a structural point of view, the shown SAML model is composed of eight main components which are Temperature Sensor, Pulse Rate Sensor, Oxygen Sensor, Sweating sensor, Vibrations Sensor, Controller, Server, and Actuator.

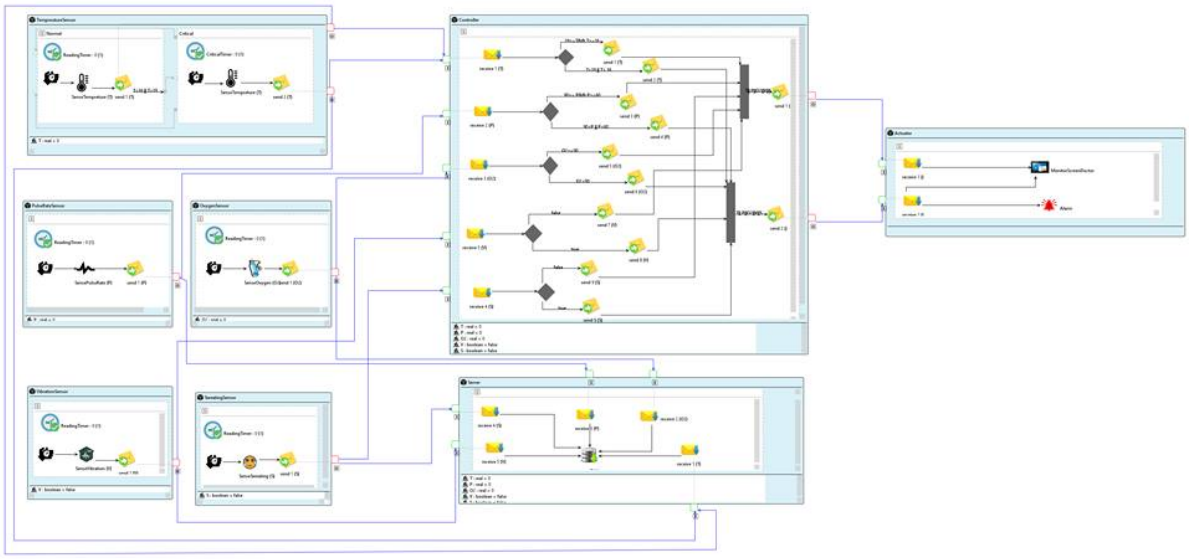

Figure 3: SAML Smart hospital gown[2]

Temperature Sensor: This component is responsible for measuring the temperature of the patient and it consists of 2 modes. In the normal mode the temperature of the patient will be measured every 1 second, this value is sent to the controller and to the server by message through the output message port, then if the tempera-

ture of the patient is less than 36 degrees Celsius or more than 39 degrees Celsius will send this reading by a message to the critical mode, which will also measure the temperature of the patient in the dangerous situation every second, then send the value to the control and to the server by message through the output message port, will be moving from critical mode to normal mode in case the return temperature of the patient to the normal temperature which is the temperature above 36 degrees Celsius and less than 39 degrees Celsius.

Pulse Rate Sensor: In this component, the heart rate of the patient is measured every 1 minute, the value will be sent by message to the control and to the server through an output message port.

Oxygen Sensor: In this component, the patient's breathing rate is measured every 1 second and the value will be sent by message to the control and the server through output message port

Sweating sensor: In this component, the patient's sweating rate is measured every 1 second to detect that if the patient was suffering from sweating and the value will be sent by message to the control and to the server through the output message port

Vibration Sensor: In this component, the patient's vibration rate is measured every 1 second to detect that if the patient was suffering from a cold and the value will be sent by message to the control and to the server through the output message port.

Controller: In this component, all values received by a received message through in port massage are checked where the temperature is measured. If the temperature is greater or equal to 36 degrees Celsius and smaller or equal to 39, then the patient is in normal status, else If the temperature is greater than 39 degrees Celsius or smaller than 36 degrees Celsius in this case the patient is in critical status.

The heart rate of the patient is also received by message through in port massage and is examined if the value is greater or equal to 60 and smaller or equal to 90 pulses in the minute, the patient in normal status either that the number of beats in the minute less than 60 minutes or more than 90 minutes, the patient in critical status.

The patient's breathing rate is also received by a received message through in port massage and is checked if the value is greater or equal to 90 in normal status. If breathing is less than 90, the patient is in

critical status.

The value of the sweating rate is also received through in port massage and is examined within the control. If the value is true, then the patient suffers from sweating. If the value is false, the patient is well status and does not suffer from sweating.

The vibration rate is also received by the massage through in-port massage and is checked inside the control. If the value is true, then the patient suffers from chills or colds. Else if the value is false, the patient is well status and does not suffer from any chills or colds.

The values of the dangerous state are sent after they are grouped by joining in a sent message through out port massage to the actuator as well as for the natural values.

Actuator: In this component, the values are received from the controller where the natural values are displayed and monitored by a screen. The dangerous values are also displayed on the screen, but with an alert to the doctor.

Server: In this component, values are received and stored.

Saml Intraocular pressure (IOP, Glocoma) Case Study

We will simply represent example of Intraocular pressure [4] case study.A simple scenario describes the monitoring of patient eye and IOP level in a patient eye. Figure 5 shows the SAML model of the CAPS. It is important to note that this figure is actually a screen-shot of our Model.From a structural point of view, the shown model is composed of four main components; IOPSensor, Controller, eyeglass Actuator, Doctor, ,and Cloud Server.

. The IOPSensor component is responsible for monitoring the the patient eye .It includes two modes: **(1) Normal mode:** in this mode the IOP sensor reads mean white water percentage in the patient eye every 604,800 sec.A timer is set in this mode to schedule IOP reading Sensor.A message carrying an IOP value is sent from the output message port of the IOP Sensor component to the import of the Controller component.Moreover, if the reading of IOP is more than 18, that means the state of the patient eye will enter the observe mode. **(2) observe mode:** in this mode the IOP Sensor reads Intraocular pressure (IOP) in a patient eye every 14400 second, since this mode Indicates the unsafe level of IOP. Also in this mode, a timer is set to schedule the reading from the IOP Sensor. A message carrying the IOP value is sent from the output message port of the IOP Sensor component to the import of the Controller

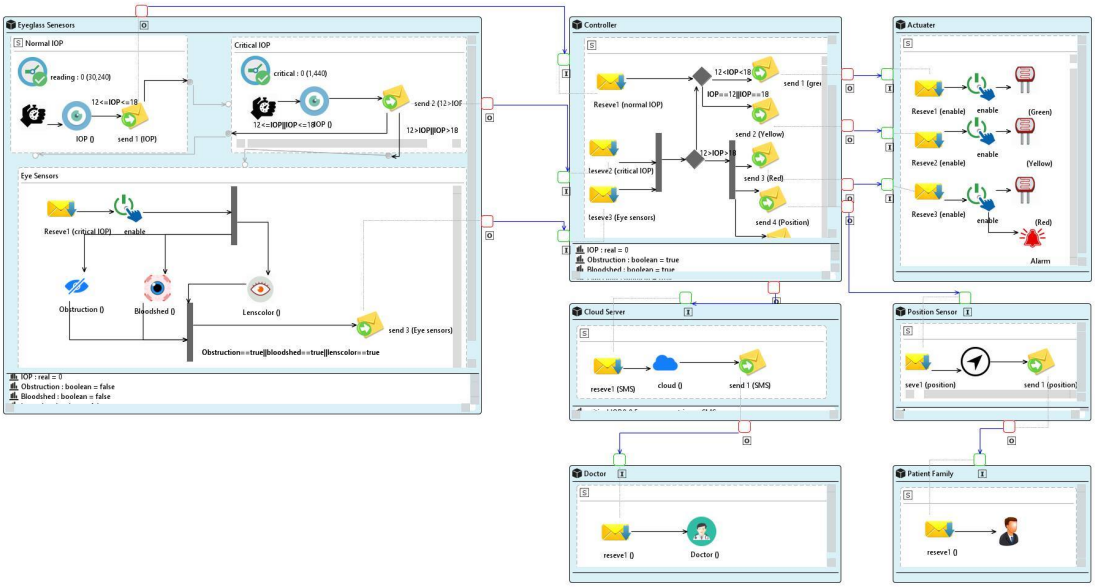

Figure 4: Saml Intraocular pressure (IOP, Glocoma) Case Study

component then a message carrying the IOP value is sent from the exit mode of the observe mode to the enter mode of the critical mode . Moreover, if the reading of IOP is less than or equal 18 or more than or equal 12 that means the state of the patient eye will go back to the normal mode. **(3) critical mode:** this mode is responsible for enable start check eye sensor value. When this mode receives an active message to the enter mode message, the eye sensors actuator is enabled for 3600 seconds to allow the entry. simultaneously, this event will distributed for 3 sensors: **obstruction sensor :** check the blurred vision . **bloodshed sensor :**check the eye congestion. **Lenscolor sensor :** check the lens of eye if redness or not. will check this values if one or more the value is true then it will send a critical message through the output port of this component to the input port of the Controller component.

The Controller component is responsible for receiving sensors data in eyeglass and take decisions based on its values. The decisions in this example are related to send a control messages to the actuators to open and/or close, led and alarm. Message contains intraocular pressure percentage: if the received value is less than 18 or more than 12 this means the patient eye in safe mode, then, this component sends through its out port a message to the

in port of the Eyeglass Actuator component. This message contains a Boolean variable (open led Green), the value of this variable is set to true. Otherwise, if the received value is equal 12 or equal 18 this means the patient eye in warning mode , this component sends through its out port a message to the in port of the Eyeglass Actuator component. This message contains a Boolean variable (open led Yellow). Otherwise, this means the patient eye in critical mode, this component sends through its out port a message to the in port of the Eyeglass actuator component. This message contains a Boolean variable (open led Red) enable alarm, sends through its out port a message to the in port of the CloudServer Component.

CloudServer Component: sends through its out port a message to the in port of the Doctor Component for continue his situation.

SAML Diabetes mellitus Case study.

We will simply represent a partial example of a Diabetes case study. A simple scenario describes the monitoring of IDiab [1] sensor reader and glucose level in patients blood. Figure 4 shows the SAML model of the CAPS. It is important to note that this figure is actually a screen-shot of Our Model. From a structural point of view.

The Model of Diabetes mellitus is composed of nine main components: IDiab Sensor, Weight of patient, Controller, Dose component, Position Sensor, Cloud Server, Patient Family Component, Doctor, and Emergency Component.

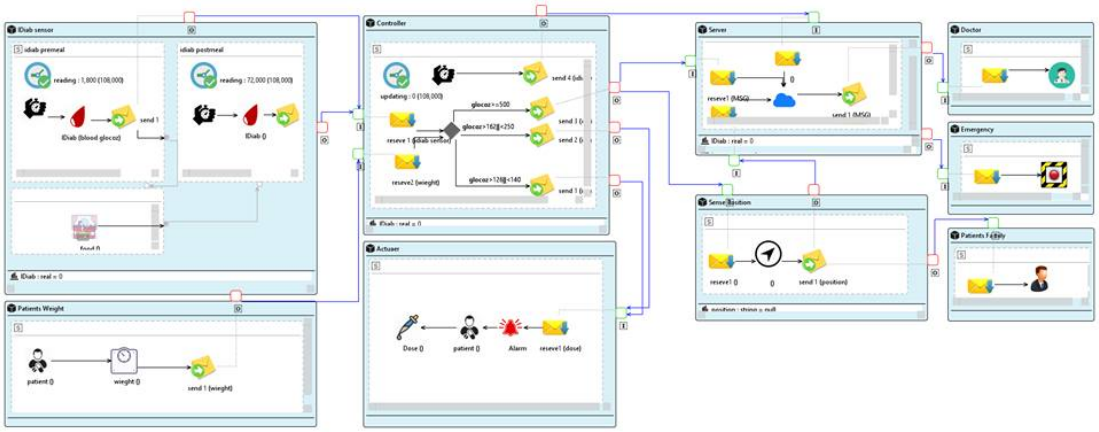

Figure 5: SAML Diabetes mellitus

The IDiab sensor Component: is responsible for monitoring the glucose percentage in patient blood. It includes two modes:

(1)Premeal mode: in this mode, the IDiab sensor reads glucose percentage in a patient blood every 14400s, we will check this value half an hour before eating. A timer is set in this mode to schedule the reading from the IDiab sensor. A message carrying the IDiab value is sent from the output message port of the IDiab Sensor component to the in the port of the Controller component. Moreover, if he was eating the food then the state needs will enter the Postmeal mode. **(2) Postmeal mode:**in this mode, the IDiab sensor reads glucose percentage in a patient blood every 14400s, we will check this value half an hour after eating. Also in this mode, a timer is set to schedule the reading from the IDiab sensor. A message carrying the IDiab value is sent from the output message port of the IDiab Sensor component to the in the port of the Controller component.

Weight patient component: is responsible to check the weight of the patient and sent the value from the output message port of the Weight component to the in the port of the Controller component which is responsible for receiving sensor data in simulation and take decisions based on its values. The decisions in this example are related to send control messages to the patient to take a number of dose for a patient that must take. In this example, the Controller component receives two types of messages from its outports: **(1)** The message contains glucose percentage in patient blood: **a-** if the received value of Premeal is between 126 and 140, this component sends through its outport a message to the in the port of the Dose Component. This message contains an integer variable (number of doses that patient need). Otherwise, if the received value is more than or equal 500, this component sends through its outport a message to the import of the Server component, this message contains a glucose critical value, and sends through its outport a message to the import of the Position component ,this message contains a string value (patient position). **b-** if the received value of Postmeal is between 162 and 250, this component sends through its out port a message to the in the port of the Dose Component. This message contains an integer variable (number of doses that patient need). Otherwise,if the received value is more than or equal 500, this component sends through its out port a message to the import of the Server component, this message contains a glucose critical value, and sends through its outport a message to the import of the Position component ,this message contains a string value (patient position). **(2)** The message contains patient weight: we need this value just once, to determine the total number of dose that the patient allows to take it every day.

The Does component: is the one responsible for alarm the patient to take a dose.

Position component: this component responsible to determine

the patient position, and simultaneously sends through it out port a message to the in the port of the Patient family Component and to Cloud Server Component and the Cloud Server component: this component receives two types of messages from its outports: **(1) IDiabsensor data**, to send this value to the doctor to take the first decision to save the patient life as much as possible. **(2) Patient position**, to send this value to the Emergency component to be taken to the hospital.

3 Simulator Diabetes mellitus

Simulate Insulin model by building a Website that computes numbers of insulin should a patient take after, before eating a meal, in addition, numbers of insulin need to decrease sugar in the blood, evaluate total daily doses, insulin between meals we take some consideration of Insulin:

1 unit of CHO utilizing 10-12 grams in Meal
1 unit of CHO decrease 50ml/dl of blood sugar
Normal blood =100mg/dl
1 carbohydrate contains 4 calories per gram.
Simulator Include 4 cases as follow:

Case one: Used for Diet it is resulting from the case two force us to commit in specific carbohydrates foods, we attached a link that's show the amount of carbohydrates foods.

Figure 6: Case1

Figure 7: Case2

Benefits of case one:

-Clarify the cause of increased Insulin in blood.

-If some patient needs to follow specific carbohydrate or within range of the pump.

-Know the type of food that caused high sugar Check if food is within the possible range of pumps or not.

Case two: Subsequently, its reads the percentage of CHO in blood by a sensor available under the skin. Called IDiab before and after eating the meal thru 2 hours, then iDiab produces values(pre-meal,post-meal) by using message passing to pass them to Server, subsequently. Into mobile application to evaluate numbers of doses insulin to decrease Sugar in blood, but if the value of post-meal is higher then 500,which means the patient is **coma** send the message to the server that contains alert for two sides one for an emergency,second for the doctor, by Using the feature of GPS to determine the place of the patient to send fast Ambulance to recover him,we should know also the peak of post-meal reach after 2-hours is 500 mg/dl.

Case Three: Evaluate total numbers of doses insulin, according to the weight, which means person check his weight by weight-device than,the simulator gives total daily doses, we make it specific case we create an android application to calculate the number of doses insulin in breakfast, lunch, dinner for Diabetic.

Figure 8: Case3

Case Four: Evaluate numbers of doses insulin to decrease Sugar in blood, Between meals, my simulator takes units of does form case 1 and case 2 and the result is needed doses between meals.

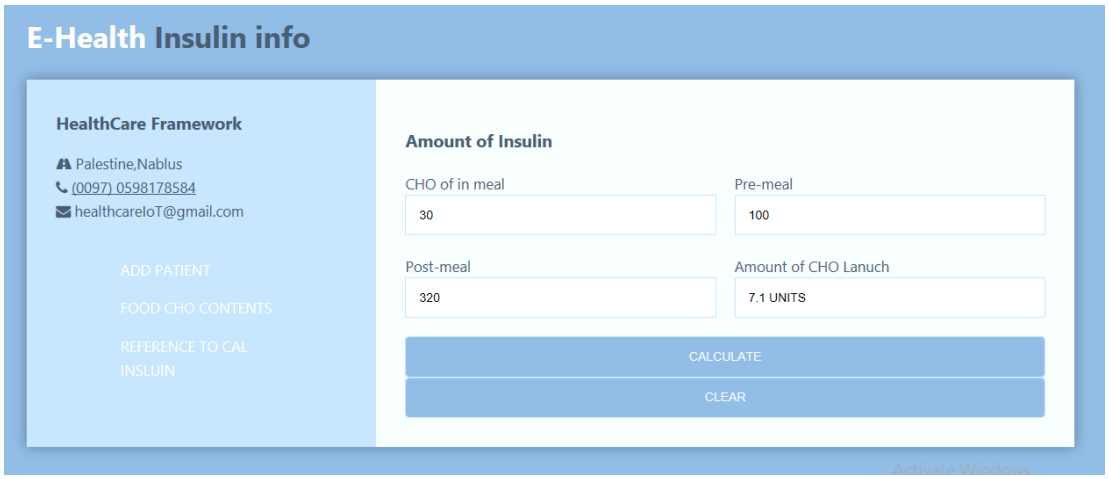

Figure 9: Case4

Generate randomly part: We bring diabetes data thru call patients and ask some questions as requested about Diabetes or conduct tests.

Which were questions as follow: **age**, **sex**, **weight**, **the average amount of Carbohydrate**, **the average amount of blood in sugar directly after wake up**, **the average amount of blood in sugar after meals**,
The goal of these data to make simulator randomly generate a results.

Age	Sex	Premeal	AverageCHO	Postmeal	#ofinsneeds	Weight	TotaldailyDoses	BetweenMeals
20	Female	105	45	180	1.5 Units	51	25.5 Units	5.6 Units
30	Female	140	60	175	1.5 Units	68	34 Units	7.0 Units
36	Female	125	45	145	0.9 Units	75	37.5 Units	5.1 Units
36	Male	85	40	140	1.1 Units	74	37 Units	4.7 Units
30	Female	100	70	150	1.0 Units	74	37 Units	7.4 Units
27	Female	120	50	160	1.2 Units	60	30 Units	5.7 Units

Figure 10: Diabetes mellitus Data

4 Mobile Application Diabetes mellitus type 1

To facilitate (Type 1 DM) pts the lives of patients Saving time for patients decrease effort to patients Increase the accuracy of the amount of insulin taken. We create this app that includes 3 stages to calculate the amount of insulin:

The first stage: calculate BMI(Body mass index) when pt. entering weight and height, and the app told you if, Underweight, Normal, Overweight, Obese.

Figure 11 shows the screen-shot of BMI Activity

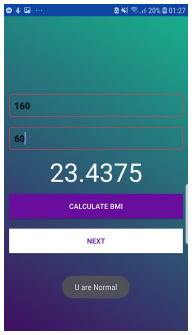

Figure 11: BMI Activity

Second stage: calculate Bolus Basel of insulin depending on your weight.

Figure 12 shows the screen-shot of Bolus Basel Activity

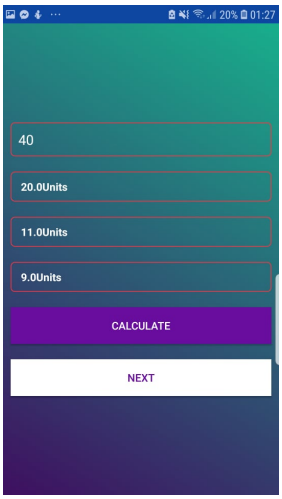

Figure 12: Bolus Basel Activity

Listing 1: Java pseudocode

```
public  void Basel&Bolus (){
        weighkg =(EditText)findViewById(R.id.weighkg);
        neededCHO=(TextView) findViewById(R.id.neededCHO);
        Bolus=(TextView) findViewById(R.id.Bolus);
        Basel=(TextView) findViewById(R.id.Basel);

    #compute the Bolus & Basel
     int value=Integer.valueOf(weighkg.getText().toString());
          Totaldailydoese= (float) (weighkg*0.5);
            neededCHO.setText("␣" + Totaldailydoese +"Units");
            float y= (float) (Totaldailydoese*0.45);
            float w =(float) (Totaldailydoese*0.55);
            Basel.setText("␣" +y +"Units");
            Bolus.setText("␣" +w +"Units");

}
```

Third stage: calculate the amount of insulin you need for this meal depends on blood glucose, U should determine first your state fasting blood sugar or random blood sugar. And the amount of carb. The result will be a fraction number and let patient determine if wants round it or not so the result means need of insulin u should take for a given meal. Usually an injection of insulin before the meal in a quarter of an hour ago. Figure 13 shows the screen-shot of Amount of insluin Activity.

Listing 2: Java pseudocode

```
public  void AmountInsluin (){

        one = (RadioButton)findViewById(R.id.Raddec);
        two = (RadioButton)findViewById(R.id.Radhexa);
       CHO=(EditText)findViewById(R.id.cho);
        Bloodsugar=(EditText)findViewById(R.id.sugar);
        totaldose=(TextView)findViewById(R.id.total);

    #compute of AmountInsluin
     float weight = getIntent().getExtras().getFloat("weight");
        rule500=500/ weight;
        rule1800=1800/ weight;
        If(patient ==FBS)
Bloodsugar=100;
Elseif(patient==RBS)
Bloodsugar=140;

            int Carb=Integer.valueOf(Carb.getText().toString());
            int Sugar=Integer.valueOf(Sugar.getText().toString());
            float correctiondose =( Sugar - Bloodsugar)/rule1800;
            float carbo = Carb /rule500;
            float dose= correctiondose+carbo;
            totaldose.setText(""+dose+"units");

}
```

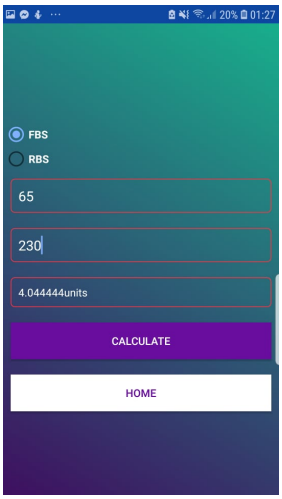

Figure 13: Amount of insluin Activity

4.1 SAML Diabetes mellitus Mobile Application

SAML Diabetes mellitus Mobile Application. We will simply represent SAML model of the Mobile Application Diabetes mellitus type 1. A simple scenario describes monitoring how Distribution of the total daily dose of a patient. We will show its SAML model. Figure 14 shows the SAML model. It is important to note that this figure is actually a screen-shot of our Model.

From a structural point of view, the shown SAML model. The Model of Diabetes mellitus is composed of six main components: BMI, Bolus and Basel, Controller, Dashboard, Actuater, AmountofInsulin.

The BMI Component: is responsible to check the weight of the patient and height in cm then evaluate of BMI from these values. sent the value from the output message port of the component to the in the port of the Controller component which is responsible for receiving data and make decisions based on its values.

The Controller Component: is responsible for receiving data in BMI and take decisions based on its values. The decisions in this example are related to send control messages to the Screen to display the message. The message contains the result of BMI message: if the received value less than or equal 18.5 this means the patient is Underweight and if value less than or equal this means the

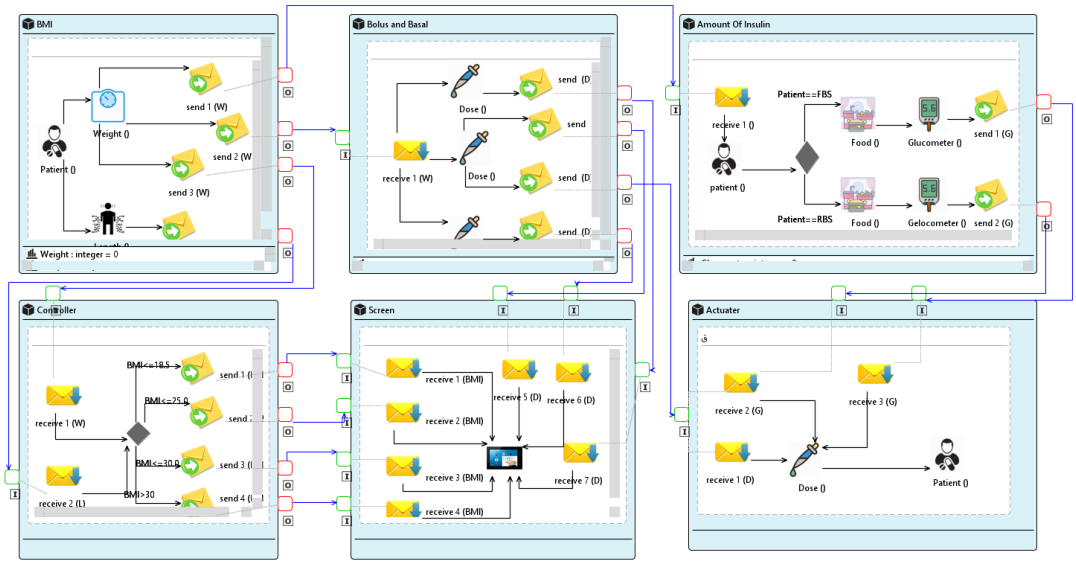

Figure 14: SAML Diabetes mellitus Mobile Application

patient is Normal, also if value less than and equal 30.0 this means the patient is Overweight ,and if the value is larger than 30.0 this means the patient is Obese. this component sends through it out port a message to the in the port of the Screen component to show it Touchscreen.

The Bolus and Basel component: is responsible for receiving weigh in BMI Component and evaluate Total daily doses and Bolus, Basel based on its value. sent the values of Total daily doses and based, bolus from the output message port of the component to port of the Screen component to show them Touchscreen and sent the value of Basel as a message from the output message port of the component to the port of the Actuater component this message contains a real variable (number of doses that patient need) to inject one to two long-injection insulin acting.

The Screen component: is responsible to display values from the input message port of receiving data from the output message port of the component BMI and receiving data from the output message port of the Bolus and Basel component.

The AmountofInsulin component:is responsible for receiving weight as message from the output message port of the BMI com-

ponent which is important factor to evaluate the amount of insulin that's injection 3 times a day with each meal based on your status if FBS or RBS and amount of carbohydrates needs each meal and the percentage in patient blood which glucometer reads it. And carrying the message whether if FBS OR RBS that contains the value of each injection insulin form output message port of the component to the port of the Actuater component.

The Actuater component:is responsible to inject patient of the amount of insulin by receiving a message that contains the number of insulin needs from the output message The Bolus and Basel component and The AmountofInsulin component.

5 CupCarbon:Energy Consumption

In this section, we will describe the results of energy consumption SAML Diabetes mellitus Mobile Application running our project in CupCarbon simulator. We applied six different components behaviours and three sensors.

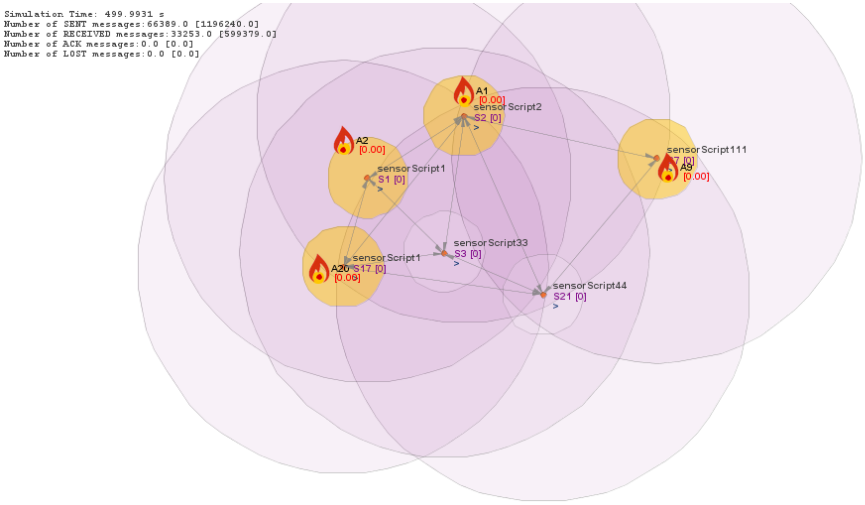

Figure 15: Cupcarbon Diabetes mellitus Mobile Application model

Figure 16: energy consumption Diabetes mellitus Mobile Application model

6 CONCLUSIONS AND FUTURE WORK

In this paper, we proposed an architecture description and associated modelling platform for the model-driven engineering E-health system thru used meta-model of CAPS framework which we customize it to make it specific for healthcare domain, CAPS would be to allow stakeholders to perform analysis for effective architectural decision making in earlier stages next to read around 150 paper related to the healthcare domain to gather all possible sensors used in this domain. We introduce a simple case study about e-health system first model is about smart hospital gown. The Smart Hospital Gown allows seamless data collection in a continuous manner in a patient-friendly manner, allowing mobility of the patient and continuous data gathering. The demonstration will illustrate the utility and practicality of the Smart Hospital Gown. The second model is about eyeglasses that use sensors in the frames to detect the eye blood pressure/flow and the colour of the lens; using an application on the smartphone, and the information is sent from the sensors to the smartphone. The application is a series of algorithms that are due to the array sensor laid out in eyeglasses, which can detect many problems of the eyes, to see if there is any kind of problem in

the eyes, and where necessary, a message will be sent to the ophthalmologist and the emergency and patient relatives; and colour of the frames turns red, and sends voice messages to the patients smartphone and makes the patient aware of the problems. Third Insulin model is about glucose monitoring system. The implemented based architecture is complete system starting from the sensor node IDiab that checks glucose for every 2 hours and send data to a back-end server. Through the system, doctors and caregivers can easily monitor the amount of glucose in the blood and the state of patient anytime, anywhere via a browser or a smart-phone application and shifting the insistence from a popular clinician-centred way patient-centred, next to build simulator and design mobile application for Diabetes mellitus type 1 .In addition, bring real-data for patients to using it to generate randomly in the simulator.

Further, it's can modelling or used e-health system to build any related health system .Also, allow stakeholders to perform analysis for effective architectural,it can be used the system academic or for research to the students and it can be for patient diabetes.

Finally, with CupCarbon simulation will help in being able to evaluate energy consumption like sensor nodes throughput. Our next plan to gain values of SAML Diabetes mellitus Mobile Application rather than manual using model transformation(Acceleo). In addition, if we bring sensors in real-life to generate them testable with Arduino code.

ACKNOWLEDGMENTS

We would like to express we special thanks of gratitude to our supervisor Dr.Mohammad Sharaf for their able guidance and support to completing our Project. We would also like to extend our gratitude to the Dr.Majdi Dwikat, DR.Qasam Asma and Dr.Intisar Al-Alem, nutrition specialist Esra Rabi, nurse Asmaa Jararaa from Palestine Diabetes Institute a for providing us with all facility that was required.

References

[1] Nicola Bui and Michele Zorzi. "Health care applications: a solution based on the internet of things". In: *Proceedings of the 4th international symposium*

on applied sciences in biomedical and communication technologies. ACM. 2011, p. 131.

[2] Maya Guru, Ragib Hasan, and Rasib Khan. "Towards non-intrusive continuous healthcare monitoring with the Smart Hospital Gown". In: *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE. 2017, pp. 618–619.

[3] Henry Muccini and Mohammad Sharaf. "Caps: Architecture description of situational aware cyber physical systems". In: *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE. 2017, pp. 211–220.

[4] Gilda Prouski, Masume Jafari, and Houman Zarrabi. "Internet of things in eye diseases, introducing a new smart eyeglasses designed for probable dangerous pressure changes in human eyes". In: *2017 International Conference on Computer and Applications (ICCA)*. IEEE. 2017, pp. 364–368.