Designing an UML-RT Model of an Insulin-and-Glucagon Pump System with Papyrus-RT

Lam Phuoc Huy

Frankfurt University of Applied Sciences, 60318 Frankfurt am Main, Germany
hphuoc@stud.fra-uas.de

Abstract. Accurate measurements and reliability are very important aspects of the modern medical device. Because the Insulin-and-Glucagon pump is a system whose failure may lead to negative consequences for the patients, it is important to ensure that the system operates correctly according to the specification. In this report, an Insulin-Glucagon pump system is modeled using Papyrus-RT. By applying the Model-Driven-Engineering concepts into the development process, the correctness of the system is assured.

Keywords: Insulin-Glucagon Pump System · Papyrus-RT · UML-RT · Modelling Language

1 Introduction

According to Johnson *et al.* [4], a medical device that is partly or fully installed into the human body and is intended to remain there is defined as an implantable device. It is estimated that 8% to 10% of all Americans and around 5% of people in developed countries equipped an implantable medical device to improve body functions and increase the quality of life [3]. An Insulin-and-Glucagon pump is an embedded system used to monitor and manage the blood sugar level of diabetes patients.

In this study, we use an open-source modeling environment called Papyrus-RT [5] to create a Unified Modeling Language Real Time (UML-RT) model of an Insulin-and-Glucagon pump system. As Baier *et al.* shown in [2], this model can help us to simulate and verify the system operation and behaviors. It is important to note that modeling is a necessary step in the development process to guarantee the correctness and reliability of the system.

The remainder of the report is organized as follows. The next section gives a glance at our project goal and objectives. In Section 3, we introduce the Insulin-and-Glucagon system description and requirements. A detailed description of our UML-RT model is described in Section 4. Finally, conclusions and future research ideas are given in Section 5.

2 Project Goal and Objectives

The goal of this semester project is to comprehend the scientific knowledge of Model-Driven-Engineering. Every student has to opt-in for an assignment's topic then design and verifies their implementation. At the end of the course, they present their learning outcomes by writing a comprehensive report and giving an oral presentation. By developing a high-integrity software application with real-world requirements, students can gain hands-on experience in all fields of software and systems engineering. As mentioned previously, we have selected to work with the Insulin-and-Glucagon pump topic. The focus of this project is on the formal modeling of the system using UML-RT modeling language. After the model is verified, it will be translated to C++ executable code for simulation-based testing.

List of tools and software our team used during the development: **Operating System**: Ubuntu 20.04 LTS; **Modeling and Code Generation Environment**: Papyrus-RT; **Modeling language**: UML-RT; **Code Management**: Github; **Diagram creation**: draw.io.

3 Case Study: Insulin-Glucagon Pump System

3.1 System Description

An Insulin-and-Glucagon pump system is a small computerized medical device used in the treatment of diabetes patients. The device is battery-operated and can be implanted into the patient's body to deliver small amounts of fast-acting insulin. The Insulin and Glucagon dosages are stored in a place called a reservoir. The purpose of the device is to mimic the way a healthy body produces Insulin/Glucagon to maintain a normal blood glucose level. This device will monitor and manage the glucose level of the patient. When the user is eating or exercising, the device will calculate the amount of insulin/glucagon needed to stabilize the blood glucose level and inject it automatically.

3.2 System Requirements

All the components of the system must be initialized successfully. The system will not be received any user input and any system failure will automatically be handled by simulation. When the simulation starts, initialized battery value and the number of Insulin/Glucagon dosages in the reservoir need to be checked first. If the value of the battery or the number of dosages is not sufficient, the system notifies the user and the simulation stops immediately.

When the system is running, the blood glucose level and the user state are shown in real-time. When the glucose level gets higher or lower than a certain threshold, the system notifies the user and automatically injects insulin/glucagon to normalizing the glucose level. If the battery or the number of insulin or glucagon dosages running out, the system alerts the user and stops the simulation. In a typical simulation, the initial battery level is at 100% and the three dosages of insulin/glucagon are stored in the reservoir.

4 UML-RT Model of an Insulin-and-Glucagon Pump System

In a UML-RT model, we have multiple components called capsules that connect with each other through ports. Capsules communicate with each other by sending trigger-based signals and the model semantics are described by the state diagrams of each capsule. Our model is configured to run with 3 threads: Top is the main thread. Top.user and Top.battery are two other threads. We chose the multi-threaded configuration because it allows our capsule to run independently. The Top capsule represents the system to be built [1]. Figure 1 shows the Top capsule of our Papyrus-RT model.

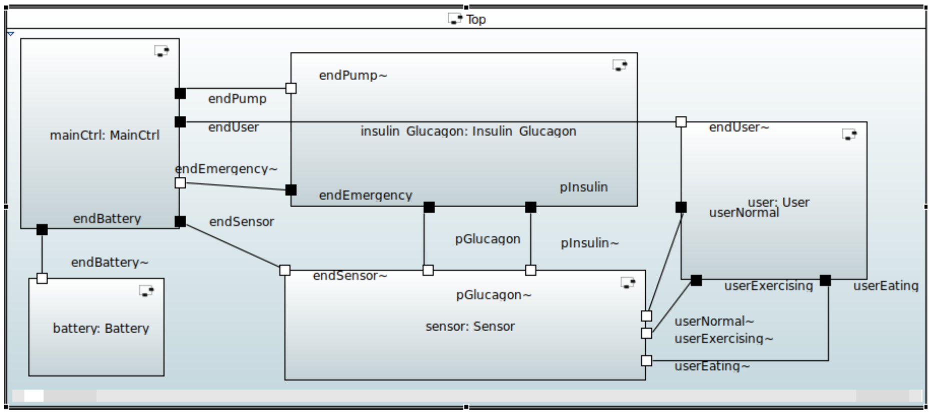


Fig. 1. Top capsule in Papyrus-RT

Battery Capsule: It simulates the battery component of the system. When the simulation begins, the initial value of the battery is checked. If the battery value is less than 15%, the capsule notifies the user and the system shut down immediately. The system will work normally if the battery value is more than 15%. We chose this threshold because the Lithium battery may start to discharge rapidly when the battery level is too low. When the system is running, the battery value decreasing 5% every 1.5 seconds. When the battery is dead, the battery capsule sends a signal to the MainCtrl capsule to shutdown the system.

Main Controller Capsule: Its job is to communicate with the Battery capsule and Insulin/Glucagon capsule. In charge of sending shutdown signal when the battery is running out or emergency shutdown due to insufficient Insulin/Glucagon dosages.

Insulin/Glucagon Capsule (Figure 2): this capsule simulates the pump and the reservoir. Its job is to inject Insulin/Glucagon into the user and monitors the number of Insulin/Glucagon dosages in the reservoir. If the reservoir is empty, the capsule sends an emergency shutdown signal to the MainCtrl capsule. We do not

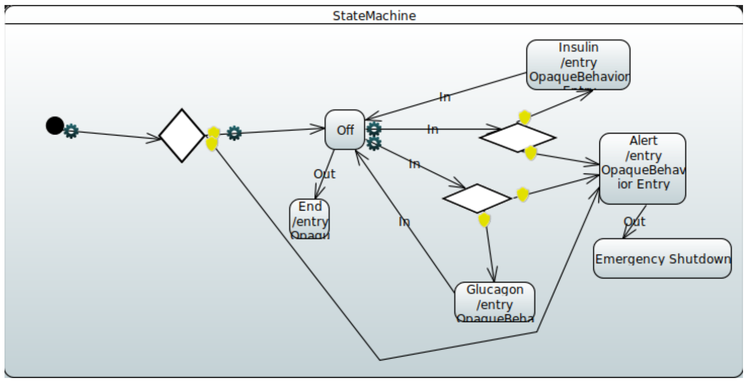


Fig. 2. State diagram of the Insulin/Glucagon component

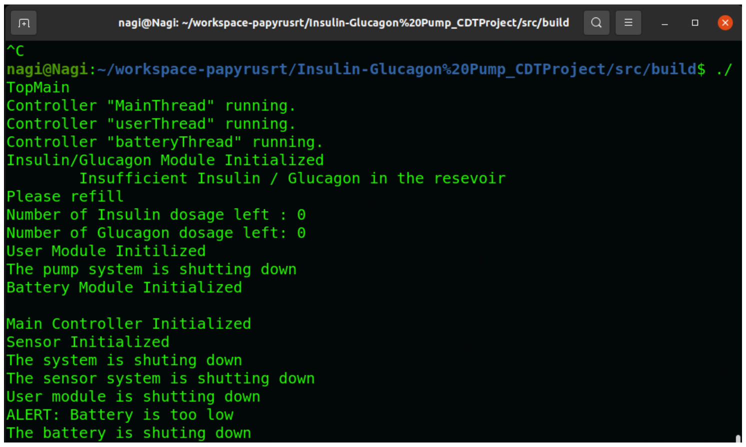
implement an algorithm to calculate the precise dosage of each injection into our model. Instead, we assume that 1 dosage of Insulin/Glucagon will be sufficient to stabilize the patient glucose level despite how many grams of carbohydrates the user is eating or how hard the user exercising.

Sensor Capsule: It simulates the sensor component. Because we don't have access to the real patient's glucose level data, we create a sensor module that produces random glucose levels based on the user state. When received information the user is eating or exercising, it sends a signal to the Insulin/Glucagon to activate the pump.

User Capsule: It defines the state of the user every 1.5 seconds and sends that information to the Sensor capsule. Our model has 3 states: normal, eating, exercising. At first, we try to generate the user state randomly. We then realize that the random algorithm in Papyrus-RT is heavily biased on some elements in the set. Moreover, it may appear that the user entering the eating state three times in a row, which does not make much sense. In the end, we chose to implement it in a pre-defined manner. The simulation starts with the user in a normal state. After some time, its changes to the eating state. After Insulin injection, it's back to normal state then finally change to the exercising state.

4.1 Running the model

Figure 3 shows our simulation run with Testcase_12. The test case scenario is to check the initial number of Insulin/Glucagon dosages. We assign the initial number of both Insulin and Glucagon dosage = 0. When our simulation is running, the system found out that the reservoir is empty. It notifies the user to refill the reservoir and shut down immediately.


```
nagi@Nagi: ~/workspace-papyrusrt/Insulin-Glucagon%20Pump_CDTProject/src/build
^C
nagi@Nagi:~/workspace-papyrusrt/Insulin-Glucagon%20Pump_CDTProject/src/build$ ./
TopMain
Controller "MainThread" running.
Controller "userThread" running.
Controller "batteryThread" running.
Insulin/Glucagon Module Initialized
        Insufficient Insulin / Glucagon in the resevoir
Please refill
Number of Insulin dosage left : 0
Number of Glucagon dosage left: 0
User Module Initilized
The pump system is shutting down
Battery Module Initialized

Main Controller Initialized
Sensor Initialized
The system is shuting down
The sensor system is shutting down
User module is shutting down
ALERT: Battery is too low
The battery is shuting down
```

Fig. 3. Running the model with Testcase_12

5 Conclusion and future research ideas

In this report, scientific knowledge of modeling and formal verification was applied to create a UML-RT model of the Insulin-Glucagon pump. In safety-critical systems like medical devices, modeling, simulating, and verifying are the necessary steps in the development process. By applying the model checking method, we can discover and eliminate hidden defects in the design early.

With the help of Papyrus-RT and UML-RT modeling language, we are able to prove that our model satisfied the pre-defined specification and working correctly. Due to the time constraint, the scale and complexity of the mentioned model are relatively small and should only be used for academic propose. Through this project, we learn a lot about Model-Driven-Engineering and gain hands-on experience in developing and verifying high-integrity software. Our plan for the future is to improve our model design to simulate and verify other scenarios that not mentioned in the test cases sheet. For example, create a scenario where the sensor provides an invalid reading value outside the normal range like 0 or negative values.

References

1. Getting started with papyrus for real time v1.0. `https://wiki.eclipse.org/Papyrus-RT/User/User_Guide/Getting_Started`, accessed: 18-2-2021
2. Baier, C., Katoen, J.P.: Principles of model checking. MIT press (2008)
3. Jiang, G., Zhou, D.D.: Technology advances and challenges in hermetic packaging for implantable medical devices. In: Implantable Neural Prostheses 2, pp. 27–61. Springer (2009)
4. Johnson, J.A.: Fda regulation of medical devices (2012)
5. Posse, E.: Papyrusrt: modelling and code generation. In: Workshop on Open Source for Model Driven Engineering (OSS4MDE'15) (2015)