Insulin Pump System Design

Introduction

The insulin pump system is a portable, automated insulin pump which is used by diabetics to administer insulin as and when they require it at regular, periodic intervals. This document illustrates the important design decisions that were made during the production of both the insulin pump software and the simulator.

Approach to implementation

The overall approach used to produce the insulin pump software was to emulate the hardware organisation by producing separate software objects (classes) for each distinguishable hardware object. Of all of these software objects the controller object is where the bulk of the computation within the system is carried. It is within the controller that the dose of insulin to be delivered is computed and where the self tests are performed. Working in together with the controller object there is a clock object which runs in a thread, constantly determining how much time has lapsed since the software was started or the timer was reset (which happens every 24 hours). Then periodically, at every interval specified the clock triggers certain events required to be performed by the system.

In order to present information to the user, a software object called display is used to create a graphical user interface (GUI), the data is then presented to the user via text boxes positioned on the GUI. The remaining software objects model the peripheral hardware units, the software contained within these objects simply records the current state of the hardware unit and for the purpose of simulation, provides the functionality to change that state.

Together the aforementioned objects combine to produce the working insulin pump system. However one other object is also provided, the simulator software object. This object runs in a thread parallel to the insulin pump software and provides the user with the functionality to perform a simulation of real-world events that would affect the pump software in differing manners. The simulator facilitates the testing process, making it quicker and easier to perform the necessary testing required in order to determine whether the insulin pump system is adequately safe.

Object Interaction – Object classes

Object classes, represented here in UML, detail the make up of the objects within the system. The objects are represented by three sections:

1. The name of the object class
2. The class attributes
3. The operations associated with the object class

Controller
Status RemainingInsulin maxDailyDose maxSingleDose minDose safeMin safeMax compDose cumulativeDose reading0 reading1 reading2
changeState(state) setDose(dose) resetCumulativeDose() updateClock(seconds) insulinReplaced() isDeliveryWorking() setDeliveryWorking(working) manualButtonPressed()

Display
mode clockTextbox outputTextbox doseTextbox manualDeliveryButton changeModeButton
Main(args) addNotify() displayDose() updateClock()

Clock
secondsElapsed
Run() displayOutput()

Sensor
sensorWorking currentReading
isSensorWorking() setSensorWorking(working) getReading() setReading(bloodSugarLevel)

PowerSupply
batteryLevelOk
isBatteryLevelok() setBatteryLevel()

Simulator
usersInput
Run()

NeedleAssembly
needlePresent
isNeedlePresent() setNeedlePresent()

InsulinPump
pumpWorking quantityToDeliver
isPumpWorking() deliverInsulin()

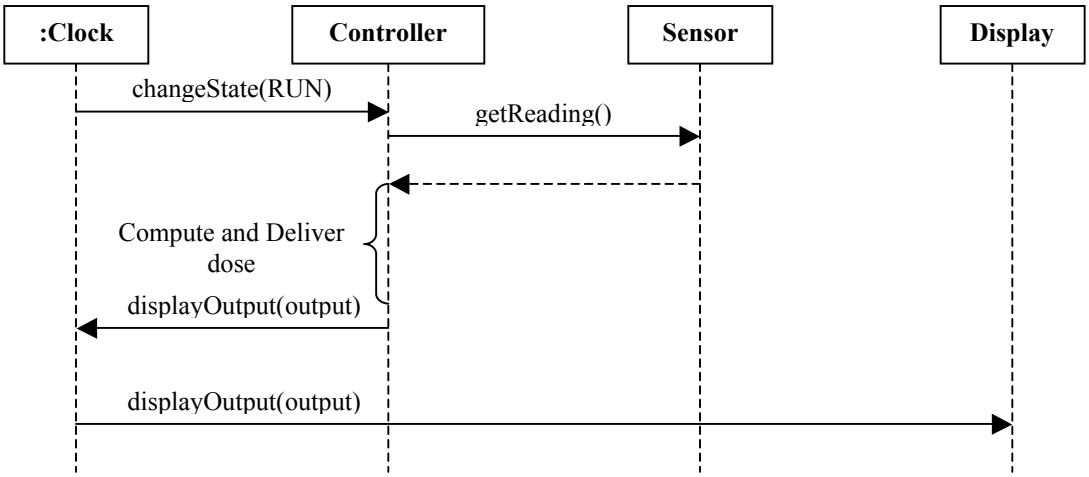
InsulinReservoir
insulinPresent
isInsulinPresent() replaceInsulin()

Object Interaction – Sequence Diagrams

In order to illustrate the sequence of object interactions that occurs in the insulin pump software the following sequence diagrams have been produced:

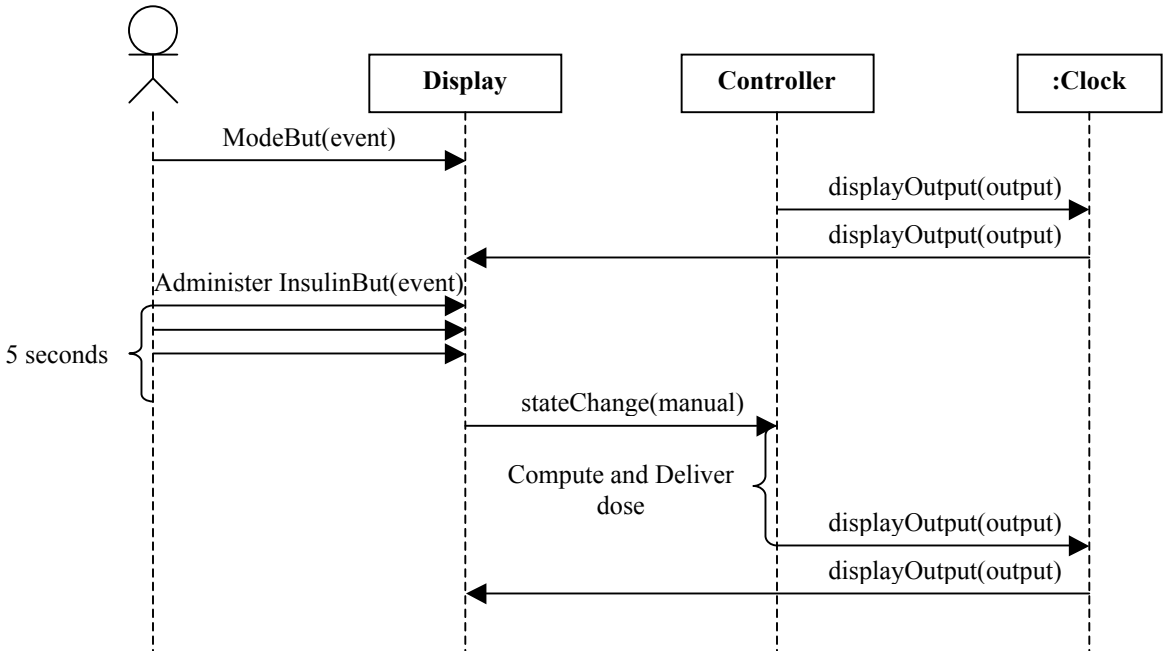
Sequence of object interactions involved in the automatic delivery of insulin

The diagram assumes that a reading has already been given to the Sensor object before the clock begins the sequence of interactions.



The final two sequences show how information that is intent to be output to the user is initially passed to the clock object where it is stored in a buffer. The contents of this buffer are then continually displayed to the user one after another for a period of five seconds each.

Sequence of object interactions involved in the manual delivery of insulin



The diagram shows that after the user has switched the insulin pump system to the manual mode they must then input how much insulin they wish to administer, this is done so by pressing the AdministerInsulinBut button as many times as is required in a five second period.

Algorithm to compute insulin dose

When the insulin pump is in automatic mode, the software periodically determines (using the blood sugar level readings) the dose of insulin that should be administered to the user. This is the functionality provided by the insulin pump software that ensures the system is considered to be a safety critical system. Therefore it is advantageous to more closely examine the algorithm that performs this task and produce a safety argument, in order to determine it is adequately safe.

Before the blood sugar level is read in, a safety check is performed to determine that the system status is correct and that the mode is set to Auto:

```
if (manDeliveryButPressed == false)
{
     if (status != ERROR && remainingInsulin >=
     maxSingleDose && cumulativeDose < maxDailyDose)
     {
```

If these requirements are met then the blood sugar level can be read in:

```
reading2 = Sensor.getReading();
```

Once the blood sugar level has been read in, the system must determine whether this level represents that the sugar level is too low, at a safe level or is too high. In order to do this the current reading is checked against the safe minimum and safe maximum blood levels defined.

Check if sugar level is low:

```
if (reading2 < safeMin)
```

if sugar level is safe:

```
else if (reading2 >= safeMin && reading2 <= safeMax)
```

or if sugar level is too high:

```
else if (reading2 > safeMax)
```

Depending on the outcome of these tests, one of three execution paths is taken, they are:

Execution path if sugar level is low

```
compDose = 0;
alarm_ON = true;
status = WARNING;
pumpTimer.displayOutput("Sugar Low");
```

If the blood sugar level is less than the safe minimum then the dose computed to be delivered is 0, the alarm is set to be activated, the status of the system is set to warning and the user is displayed a warning message reading “Sugar Low”.

Execution path if sugar level is safe

If the sugar level is deemed to be safe then the current reading is compared against the previous reading in order to determine whether, since the previous reading the blood sugar level is stable or falling, if it is then the dose computed to be administered is set to 0:

```
if (reading2 <= reading1)
{
     compDose = 0;
}
```

If this is not the case then the sugar level must be rising and a relative dose of insulin must be calculated to attempt to prevent any further increase in the sugar level:

```
// If rate of increase is falling
if ((reading2 - reading1) < (reading1 - reading0))
{
     compDose = 0;
}
// If rate of increase is increasing
else if ((reading2 - reading1) >= (reading1 - reading0))
{
     // If dose is rounded to zero, deliver the min dose
     if ((reading2 - reading1) / 4 == 0)
     {
          //set the amount to deliver to the min dose
          compDose = minDose;
     }
     else if ((reading2 - reading1) / 4 > 0)
     {
     //Set the amount to deliver
          compDose = (reading2 - reading1) / 4;
     }
```

```
}
```

Once it has been determined that an increase has occurred, the first test the software performs is to check whether the rate of increase is falling overall. If it is then the dose computed to be administered is set to 0. Otherwise, the software then proceeds to ensure that the rate is therefore increasing, if it is then a dose to be administered is computed depending on the severity of the increase. If the rate of increase is not severe enough to be rounded to higher number than 0 then the minimum dose is administered.

Execution path if sugar level is too high

If the sugar level is otherwise deemed to be too high the following code is executed:

```
// If Sugar level increasing
if (reading2 > reading1)
{
     // If dose is rounded to zero, deliver the min dose
     if ((reading2 - reading1) / 4 == 0)
     {
          compDose = minDose;
     }
     else if ((reading2 - reading1) / 4 > 0)
     {
          //Set the amount to deliver
          compDose = (reading2 - reading1) / 4;
     }
}


// If the Sugar level is stable
else if (reading2 == reading1)
{
     compDose = minDose;
}
// If the Sugar level is falling
else if (reading2 < reading1)
{
     // If rate of decrease increasing
     if ((reading2 - reading1) <= (reading1 - reading0))
     {
          compDose = 0;
     }
     // If rate of decrease decreasing
     else
     {
          compDose = minDose;
     }
}
```

Again, the software determines whether the sugar level overall is increasing, decreasing or whether the level has remained stable. The dose is calculated in the same manner as it would be had the sugar level been safe, except when the sugar level high the software not only checks whether the sugar level is falling but also checks the rate at which it's falling. If the rate of decrease is increasing then the computed dose is set to 0 whereas if the rate of decrease is decreasing, the minimum dose of insulin is computed to be administered. Also if the sugar level is too high, the software computes that the minimum dose should be delivered if the level has remained stable.

Safety of the algorithm

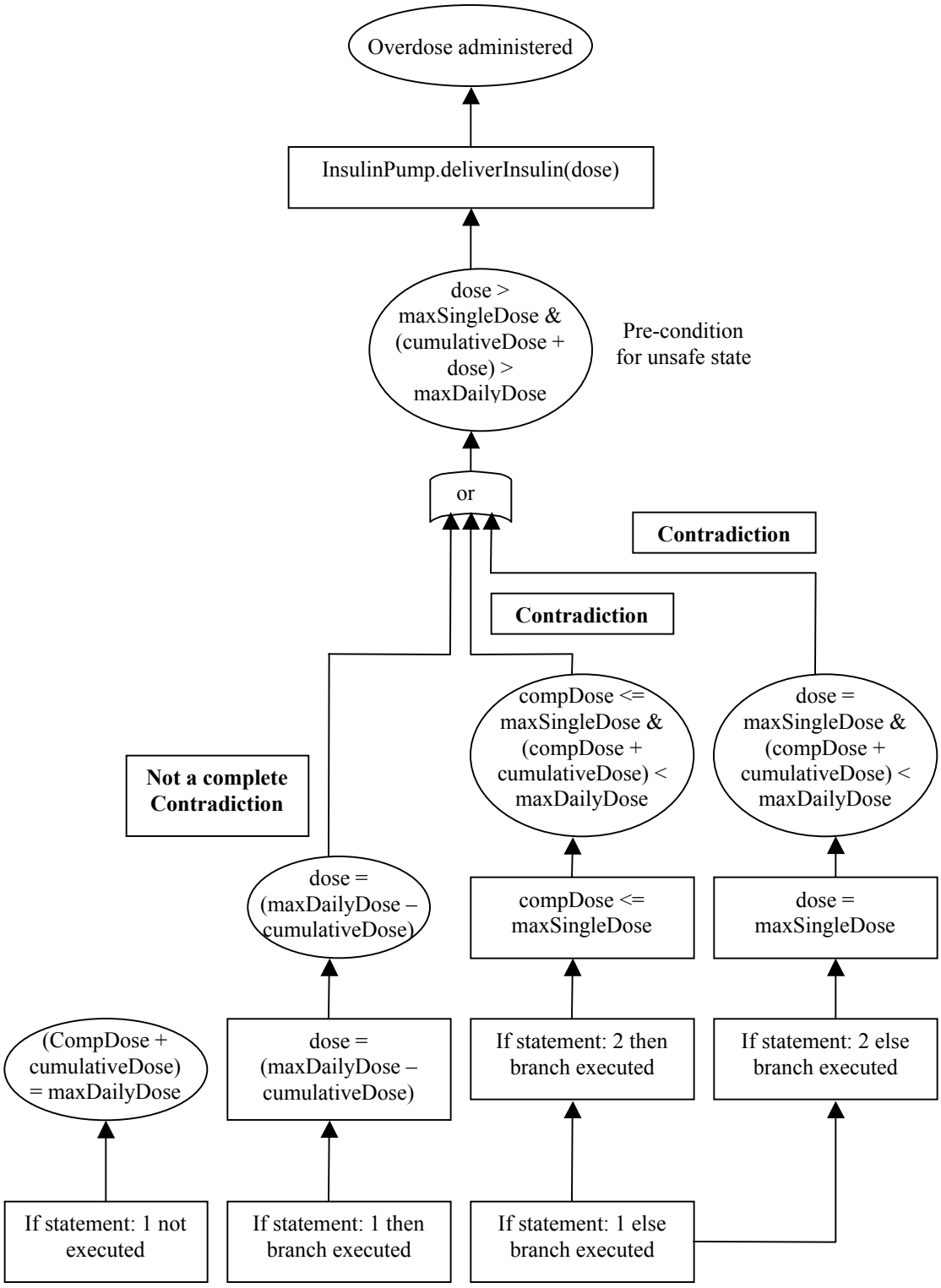
Once the dose has been computed the system then must determine whether the dose calculated is safe, this is essential due to the critical nature of the software. In an attempt to prove to a satisfactory level that the insulin pump software is safe we can use the safety argument method of validation, in order to make certain that the insulin pump system cannot deliver an unsafe quantity of insulin. Therefore the section of code we must produce a safety argument for is the following:

```
//IF STATEMENT: 1
//If max daily dose will be exceeded by the dose calculated
if ((compDose + cumulativeDose) > maxDailyDose)
{
     //Alert the user and set status to Warning
     alarm_ON = true;
     status = WARNING;
     dose = maxDailyDose - cumulativeDose;
     InsulinPump.deliverInsulin(dose);
     pumpTimer.displayOutput("Close to Max Daily Dose");
}
//Normal situation
else if ((compDose + cumulativeDose) < maxDailyDose)
{
     //IF STATEMENT: 2
     if (compDose <= maxSingleDose)
     {
          dose = compDose;
          InsulinPump.deliverInsulin(dose);
     }
     //If the single dose computed is too high
     else
     {
          dose = maxSingleDose;
          InsulinPump.deliverInsulin(dose);
     }
}
```

There are two pre-conditions that would make the system unsafe a) if the dose calculated to be delivered is greater than the maximum single dose value b) if the maximum daily dose was exceeded.

Using the safety argument we must prove that the all the if-statement post conditions contradict the unsafe pre-condition:

Safety Argument


Overdose administered
InsulinPump.deliverInsulin(dose)
dose > maxSingleDose & (cumulativeDose + dose) > maxDailyDose
Pre-condition for unsafe state
or
Contradiction
Contradiction
Not a complete Contradiction
compDose <= maxSingleDose & (compDose + cumulativeDose) < maxDailyDose
dose = maxSingleDose & (compDose + cumulativeDose) < maxDailyDose
dose = (maxDailyDose – cumulativeDose)
compDose <= maxSingleDose
dose = maxSingleDose
(CompDose + cumulativeDose) = maxDailyDose
dose = (maxDailyDose – cumulativeDose)
If statement: 2 then branch executed
If statement: 2 else branch executed
If statement: 1 not executed
If statement: 1 then branch executed
If statement: 1 else branch executed

Results

The safety argument shows that the potentially unsafe state could be reached and potentially a user could be administered the maximum daily amount of insulin in one go. This would happen if this sequence of events were to occur:

- `cumulativeDose = 0`
- `compDose computed to be a very high number`
- `If statement: 1 then branch executed`
- `dose = maxDailyDose – 0`
- `InsulinPump.deliverInsulin(dose)`

} maxDailyDose delivered all at once

Producing the safety argument has lead me to the conclusion that the insulin pump software (if implemented according to the Z specification provided) is unsafe for the aforementioned reason. The Z specification specifies that dose the dose should be administered as a post condition of If-statement: 1 (the requirement to administer the insulin at this point is modelled in the Z specification as dose!). In order to eliminate this safety error the point at which insulin is administered (dose!) would have to be moved in the specification.

Java Code

The Insulin Pump System Java code can be found at the following address:

www.lancs.ac.uk/ug/mccarthb/insulinpump.zip