# Software Design (CSE 564)

Spring 2019

**DESIGN AND IMPLEMENTATION FOR AN ISOLETTE SOFTWARE SYSTEM**

Snehit Mikkilineni Riddhi Patel Shivam Shah

School of Computing, Informatics, and Decision Systems Engineering

Arizona State University, Tempe, AZ, USA, 85281

# Table of Contents



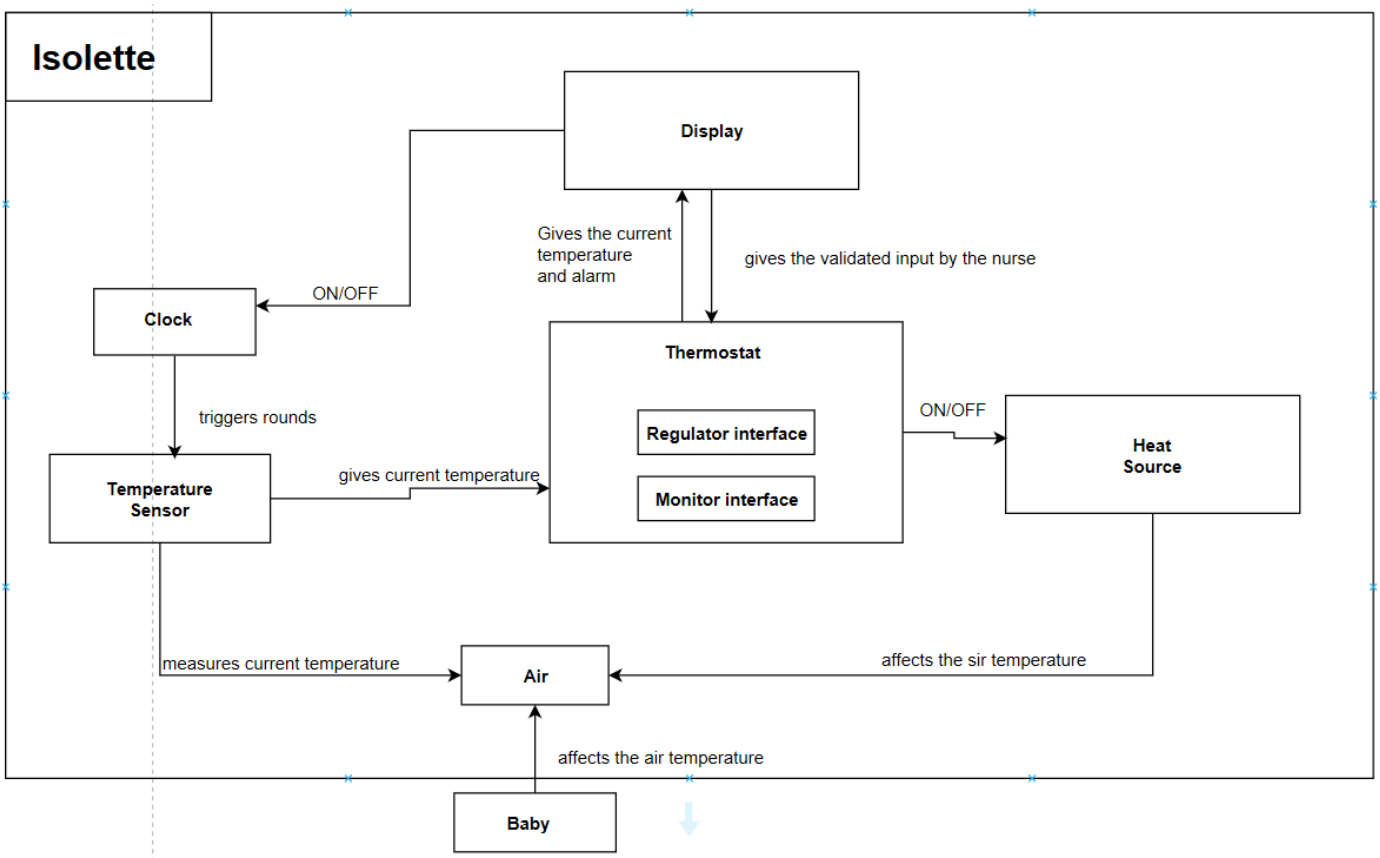
| | **Snehit Mikkilineni** | **Riddhi Patel** | **Shivam Shah** | | |
|---|---|---|---|---|---|
| **Parts** | **% Effort** | **% Effort** | **% Effort** | **Points*** | **Earned Points** |
| Problem description | 33.3 | 33.3 | 33.3 | 5 | |
| SRC design specifications with description | 33.3 | 33.3 | 33.3 | 25 | |
| UML design specifications with description | 33.3 | 33.3 | 33.3 | 25 | |
| Implementation | 33.3 | 33.3 | 33.3 | 15 | |
| Experiments & results | 33.3 | 33.3 | 33.3 | 10 | |
| Conclusions | 33.3 | 33.3 | 33.3 | 5 | |
| Demonstration | 33.3 | 33.3 | 33.3 | 5 | |
| Presentation | 33.3 | 33.3 | 33.3 | 10 | |
| Code quality | 33.3 | 33.3 | 33.3 | 5 | |
| Report quality | 33.3 | 33.3 | 33.3 | 5 | |
| Total | 33.3 | 33.3 | 33.3 | 110 | |

# Software Design (CSE 564)

## 1 Problem Description

The Isolette is an incubator system for infant, which is primarily used for providing a controlled environment with safe temperature for the infant. The main control of the Isolette system, namely the Thermostat, maintains temperature within Isolette within a safely defined range for the infant. Whenever necessary, the thermostat turns on the heat source on and off to adjust the temperature accordingly. For safety purposes, if the temperature falls out of the safe range, either too high or too low, an alarm goes off, which alerts the Nurse to reset the temperature accordingly. More specifically, when the current temperature falls out of the alarm range, that is when the alarm is triggered. Initially, when the Isolette is powered on by the nurse, the nurse actually inputs the desired temperature range and alarm temperature range into the system.

**First, let's look at the high-level design of overall system:**



## 2 DESIGN

It describes an overview of a design for the Isolette Software System [5] developed using UML, Synchronous Model, and Asynchronous Model methods.

### 2.1 SRC design specifications with descriptions

It provides SRC design specifications with descriptions highlighting the design ideas, decisions, and rationale [1].

**Initially, let's start with the smaller components contained in our SRC diagram.**

**Baby:**

-In our Baby SRC component, the two input channels are event second, and a boolean babyStatus variable.

-First, the first input into the component is that event second input. This is an event which is received from our Clock component, which will be discussed later. That event received from Clock is what synchronizes the components within each round.

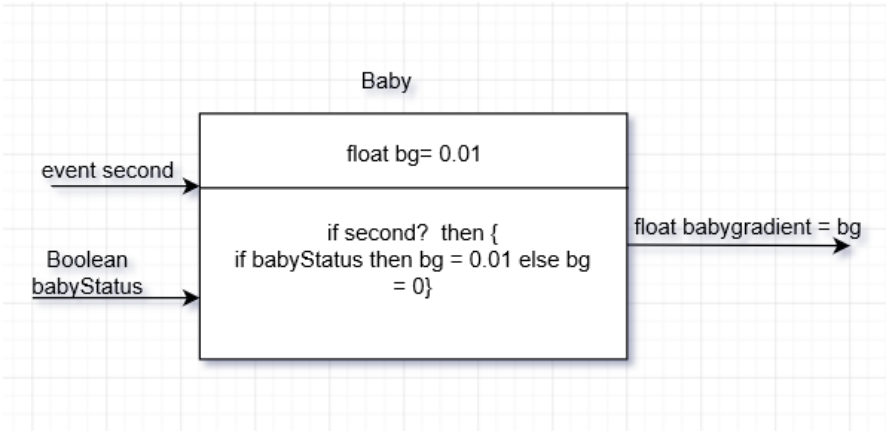
-Secondly, the babyStatus boolean variable is a variable that simply keeps track of whether the baby is in the Isolette, or if it has been removed from the Isolette.

-Then, that babygradient output variable is the gradient that takes into account the heat that is absorbed by the baby while in the Isolette. So that is the reasoning behind the gradient being outputted from the Baby component. In the overall process of calculating the current temperature of the Isolette environment, this babyGradient is going to be added to other values and gradients in order to calculate the temperature.

Assumptions:

1. Baby will always contribute to the heat gradient by way of body temperature. (babygradient variable)
2. When thermostat displays message to Nurse to put the baby inside, the nurse puts baby inside the Isolette. So in other words, it is assumed that the Nurse will follow the directions of the Isolette system.

-The component SRC diagram for the Baby is displayed below:



**Air:**

-In our Air component, there are two inputs, babygradient and heatgradient. And there is one output for roomtemp.

-Firstly, the babygradient input into this component will be the output from the Baby component. This gradient basically takes into account the heat given out by the Baby into the environment in the isolette.

-Next, the heatgradient input into the component will be output from the HeatSource component. This is the specific amount which will either be incremented or decremented from the current temperature in Isolette, depending on whether the HeatSource is on or turned off.

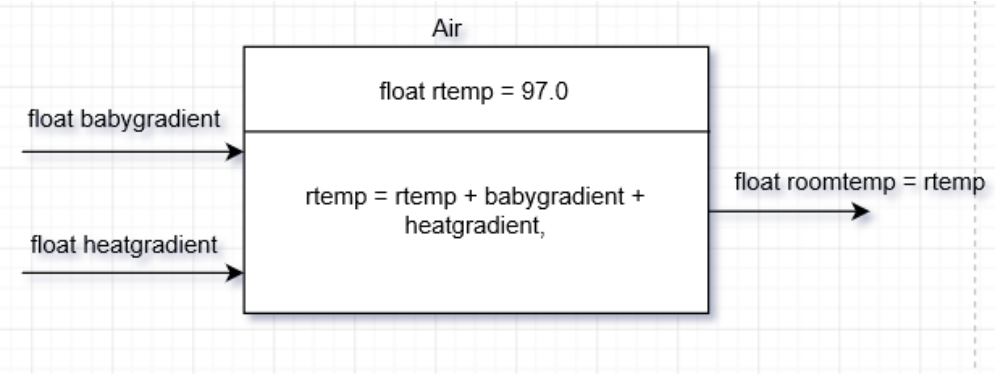
-So the reasoning behind the two gradients, babygradient and heatgradient, being input into the Air component is that the Air component, in a way, symbolizes the Air in Isolette, and so it will have heat input into it (HeatSource On) or taken out of it (HeatSource Off) and will consequently affect the Air.

-Lastly, the output of this Air component will be the roomtemp.

Assumptions:

1. Both input gradients are consistently updated and passed into Air component.

-The component SRC diagram for the Air is displayed below:




**Clock**:

-The reasoning behind us creating a Clock component was for there to be a component which could act as a coordinator for all the other components and synchronize them.

-In our Clock component, there are two inputs, event on and (real time) second, and there is one output, event second.

-Firstly, the event on input was created in our Clock component, because we needed an event which signifies that the Isolette is turned On. The Nurse turns on the Isolette, and that triggers the event on switch for the Clock component, So whenever that is triggered, the Clock will start running, and in turn, all components will start executing.

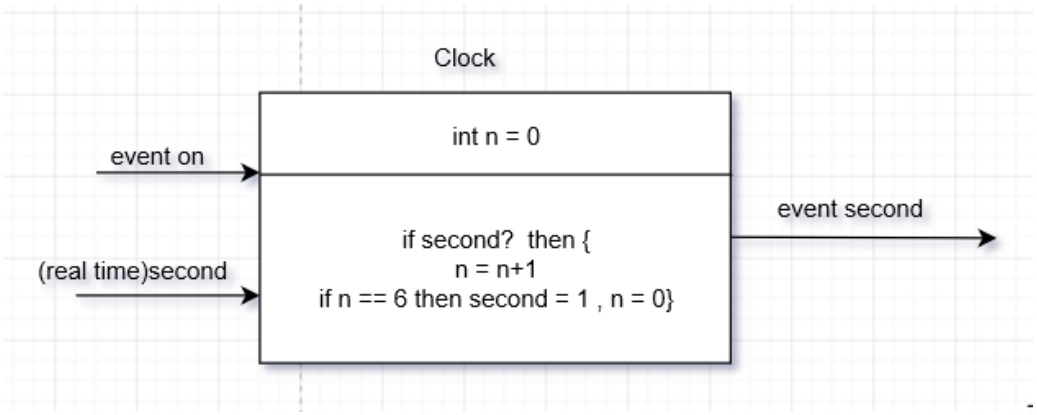
-Next, the (real time) second is the second input into the component.

-Lastly, the event second output was specified for the component, because this is the event which synchronizes all the components of Isolette within each specific round of execution.

Assumptions:

1. Every 6 seconds, event is triggered and a new start of a round is starting.

-Below is the component for Clock:




**HeatSoure:**

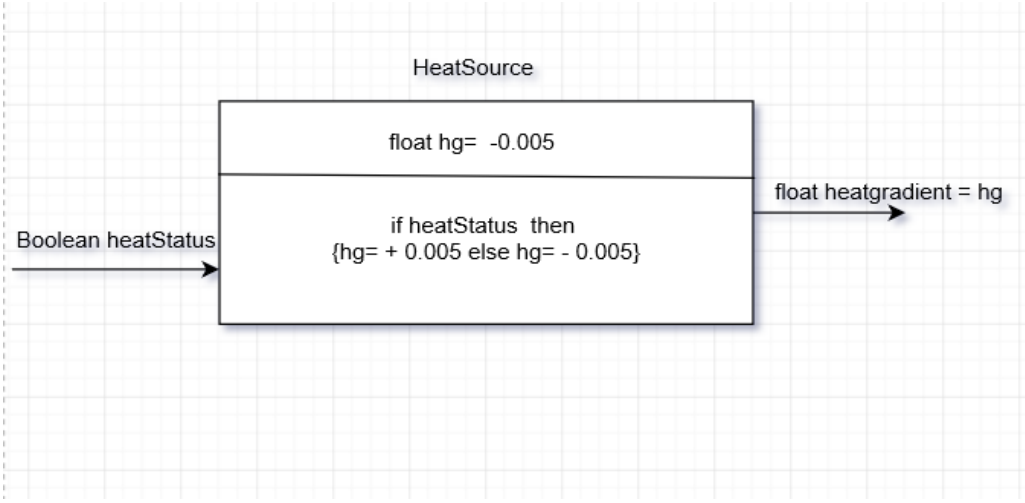
-The input into the HeatSource component is just the heatStatus boolean variable. This variable states whether the HeatSource is turned on or not. The thermostat is the component which turns the HeatSourse On and Off.

-The output for the component is the heatgradient, which is the value by which the current temperature will change depending on whether the HeatSource is on or off.

Assumptions:

1. The heatStatus will be received from the Thermostat component continuously throughout execution, so that the HeatSource will always know when to turn On and Off.

-Below is the diagram for the HeatSource component:



# Software Design (CSE 564)

**TempSensor:**

-The input into the TempSensor component will be the roomtemp. The outputs are tempSensorStatus and curr_temp.

-Firstly, we have specified the roomtemp input for HeatSource because the TempSensor needs to take into input the current temperature of the environment in order to output the value of the temperature to the Thermostat.

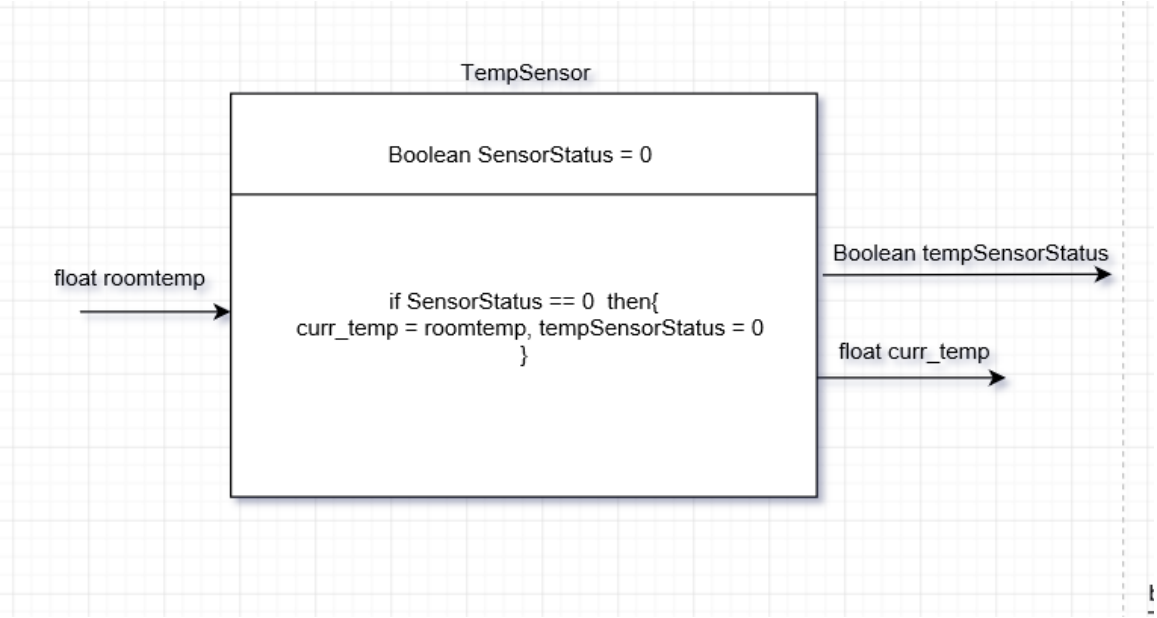
-Next, we specified the tempSensorStatus output variable for the component in order to check for internal hardware failure of the Isolette.

-Lastly, the curr_temp output variable is there for outputting the current temperature and sending to Thermostat.

Assumptions:

1. The tempSensorStatus variable will always be false, as we are assuming there will be no failures in the system in regards to Hardware.

-Below is SRC diagram for TempSensor:



**Thermostat:**

-The inputs into Thermostat component are the following: curr_temp, alarm_max, alarm_min, bool n, desired_min, desired_max, tStatus, mStatus and tempSensorStatus. The outputs are the following: mode, curr_temp, babyStatus, heatStatus and alarmStatus.

-We specified the curr_temp as an input so that the thermostat can have the value of current temperature in order to update statuses accordingly.

-We have the alarm and desired range min and max variables inputted into this component so that the Thermostat can use those ranges in order to update the regulatory and monitor statuses.

-We have the tStatus boolean variable and mStatus boolean variable in order to maintain boolean values for the monitor and regulator statuses.

-Next, we have specified the tempSensorStatus boolean input to keep track of any possible hardware failures. However, in our implementation, we have made the assumption that there will be no issues with hardware.

-Lastly, the boolean n input is there so that we can retrieve the thermostatMode and use it in the conditional logic within Thermostat component.

-Now, the first output, boolean mode, was specified so that the thermostat mode can be outputted through this component. This thermostat mode is used in order to see if the thermostat is running in NORMAL mode or if it is in FAILURE.

-Next, the curr_temp output is specified so that the thermostat can send it to Display component in order to display the current temperature.

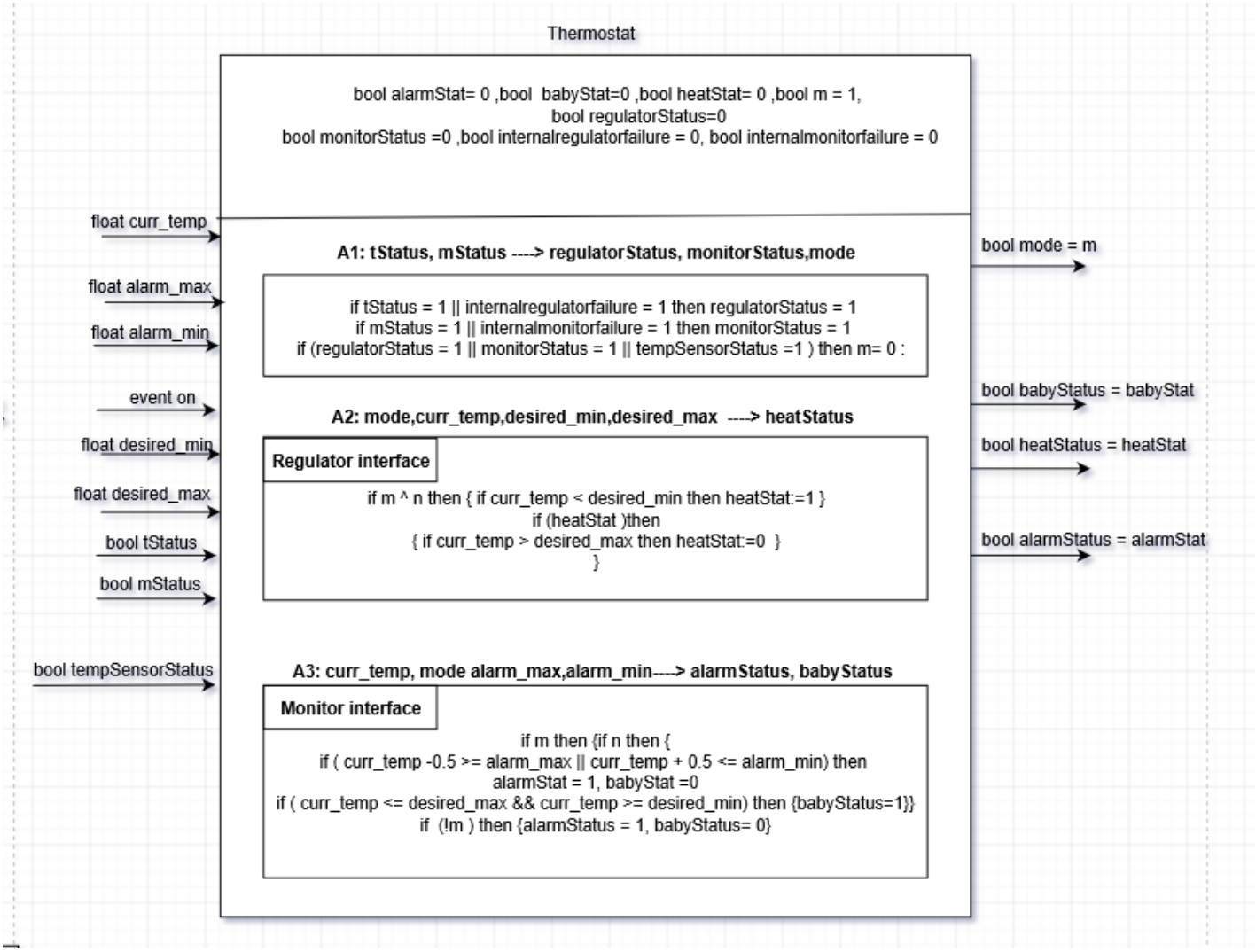
-The babyStatus boolean variable is output from the component because the status is adjusted within the logic of Thermostat component accordingly, based off of the current temperature and where it is in relation to desired and alarm temperature ranges.

-The heatStatus and alarmStatus values also are changed within logic of Thermostat component and are accordingly given out from component, so that we can tell if heatSource is On or Off and see whether alarm id On or Off.

**NOTE:** Within this Thermostat component, of which the diagram is displayed below, the subtask A1 will run first, and then only will the subtasks A2 and A3 run concurrently. This is so that the Regulator Interface and Monitor Interface run concurrently as threads.

Assumptions:

1. tempSensorStatus will always be false (we are assuming there will be no hardware failures)
2. The curr_temp input value will have the same value as the curr_temp output value for the Thermostat component.

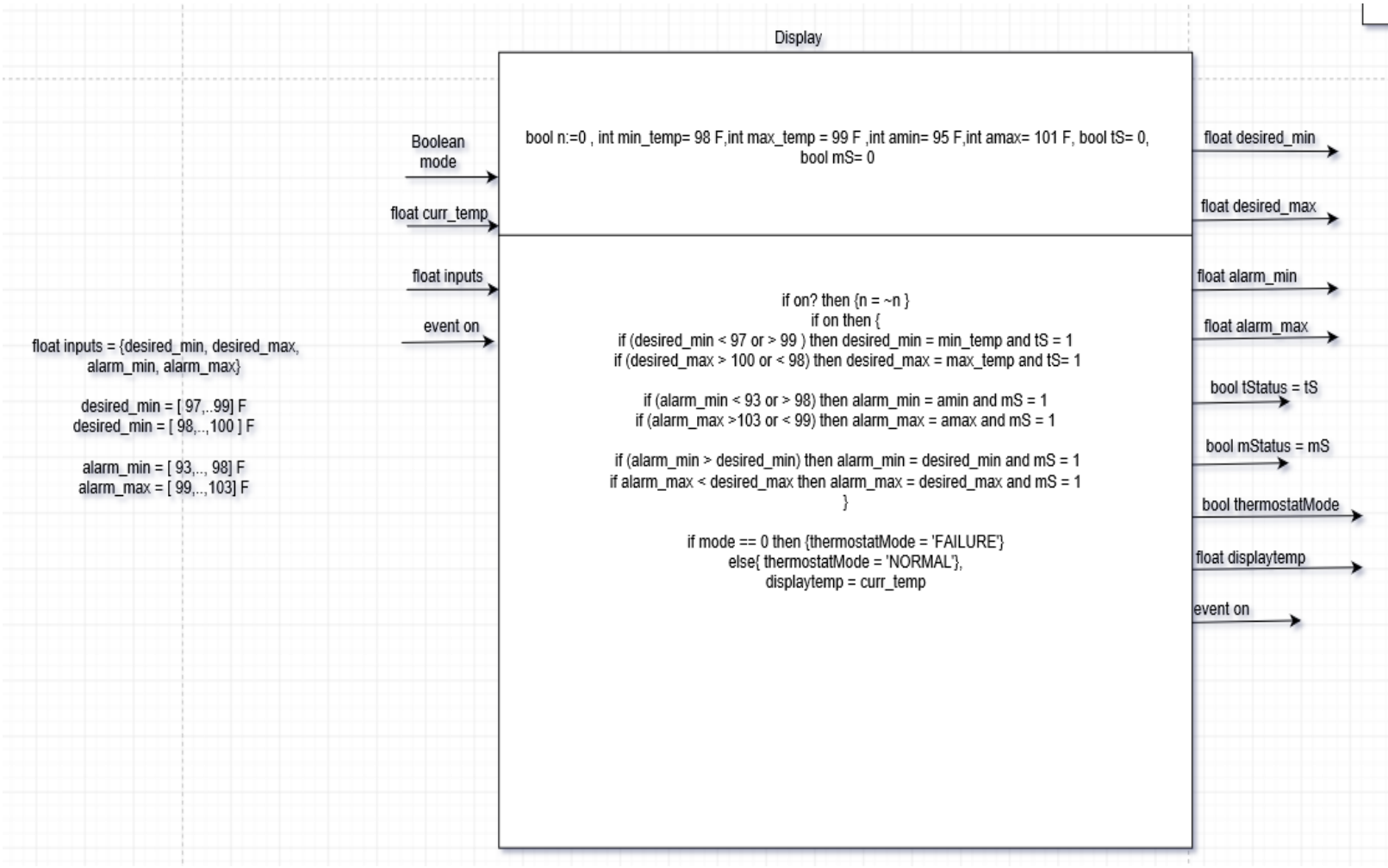



**Display:**

-The inputs into this component consist of the thermostat mode, the current temperature, the event on sent from Clock, and float inputs, which correspond to the desired and alarm temperature ranges.

-In the outputs of component, some of the notable are related to the exact temperature ranges of both the alarm and desired temperature ranges.

-The internal logic of the Display component deals with logic for setting tStatus and mStatus, which are addressing statuses corresponding to whether the inputted ranges from nurse are valid or not.




## 2.2 UML design specifications with descriptions

UML designs (Unified Modelling Language) is a modelling language used to visualize the system and the various components of the system. It helps us to develop the system in a much more modular manner and also makes easy for the engineers to maintain the overall system. For that manner we can also say that when there is a change in a particular component of the system we don't have to change the entire system as the other components only interact with them via their outputs and associations. Hence another aspect of UML is providing abstraction **[2,4]** to the system.

Keeping the advantages of UML designing in mind we use the Class Diagrams to draft a design for the Isolette.
It is one of the main building blocks of the software designing process.

By analyzing behavior and type of structure of various objects of the system we have formed the classes. Real world like tangible entities with similar behavior and structure have been kept in the same class. Each class has attributes and methods which are implemented for the desired execution of the Isolette.

Following are the classes which we have formed for the better understanding of our implementation purpose:

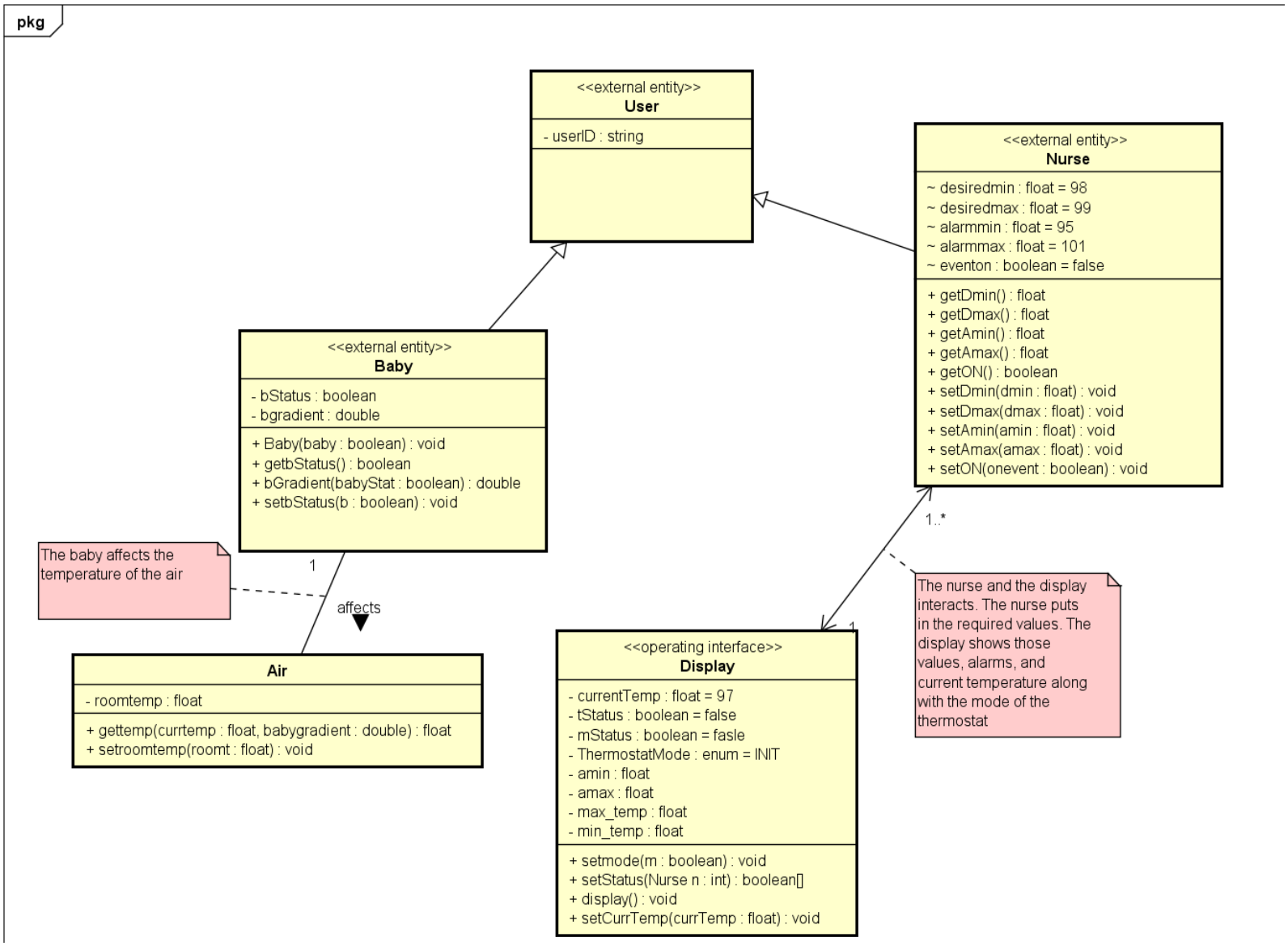
# Software Design (CSE 564)

1. **Baby**

The class Baby is the class affecting the class Air. Baby is to be kept safe is the main motive of the entire system. When the baby is put inside the isolette it starts affecting the temperature of the closed isolette. Hence the main implementation the Baby is to check when the baby is inside the isolette and if the baby is present then it contributes towards increasing the temperature of the Air.
Baby is an external entity placed inside the system and acts as an actor affecting the components of the system.

Baby *'IS – A '* type of user of the system.

The bStatus represents the presence of the baby and get, set methods are there in the class for receiving and setting the bStatus value. This status is further used by the bGradient method to calculate the value of the heat dissipated by the baby in the Air.



**2. Air**

Air is the main component which is sensed, and its temperature is to be maintained so as the baby is comfortable in the isolette.

The Air component is *associated* with the Baby for taking into consideration the increase in temperature when baby is inside the isolette. It is also *associated* with Heat Source component of the isolette. The HeatSource also contributes towards increasing and maintaining the desired temperature of the Isolette.

Air has the roomtemp which is the initial room temperature of the Isolette which we have passed from a file for the sake of execution. We can change the initial value to mimic the real world scenario. The gettemp() is used to calculate the increase/decrease in the temperature of the air due to the heatsource and the baby.

**3. Nurse**

The Nurse initiates the entire process and the system. Keeping the temperature constraints in mind the Nurse interacts with the Display of the Isolette. She enters the desired temperatures ranges which are suitable for the Baby (on the basis of the weight and health of the baby). There is a two-way communication (*association*) between the Nurse and the Display class as the Nurse also receives information from the Display class. Nurse also puts in the ranges for when the Alarm should start ringing.

There are get, set methods for all the inputs by the Nurse in this class.

Nurse *'IS – A '* type of user of the system and she has *1-1 interaction* with the Display class.

**4. Display**

This class acts as an *operating interface* of the system. The setStatuses() validates the ranges of the inputs by the Nurse. The validation is done on the basis of the constraints mentioned in the requirements of **[5].** It also checks for the mode of the overall Isolette in the setMode() i.e internal failures any of the component. It displays the current temperature of the isolette continuously to the Nurse.

The display '*is a part (aggregation)'* of the isolette. It can exist as an independent component also. For every isolette there is one Display.

**5. Alarm**

Alarm '*is a part (aggregation)'* of Thermostat class. It is notified by thermostat when to display the Alarm. For every Thermostat there is 1 Alarm.

**6. Thermostat**

Thermostat is the component consists of two sub parts:

**Monitor Interface**

The monitor interface looks after the alarm range. Whenever the current temperature given by the tempSensor goes above or below the alarm range which is input by the Nurse than it triggers the Alarm class to execute.It warns the Nurse by executing the setStatuses().

**Regulator Interface**

The main function of the setHeatSource() is to turn ON and OFF the heat source by setting its status in the Heast Source class. When the current temperature falls below the desired minimum value it sets the heat Status which in turn turns the Heat Source ON. Similarly, when the current temperature rises above the desired maximum value it sets the heat Status to false which in turn turns the Heat Source OFF.

Besides this Thermostat also sets the working mode of the system to INIT, FAILURE and NORMAL mode according to the validations in **[5]**.

Thermostat '*is a part (aggregation)'* of the isolette. For every isolette there is one Thermostat

7. **HeatSource**
   Heat Source gives out the heat gradient in incrTemp(). The get, set methods for the heatStatus are used to receive and give the heatStatus by Thermostat and Air classes respectively. When the heat status is set by the Thermostat then the Heat Source is turned ON and gives a heatgradient of +0.008 to the Air to increase its temperature. When thermostat sets the value of heatStatus in Heat Source as false then the Heat Source is turned OFF. While Heat Source is OFF the heat is absorbed hence, cooling down the temperature of the Air by -0.08.

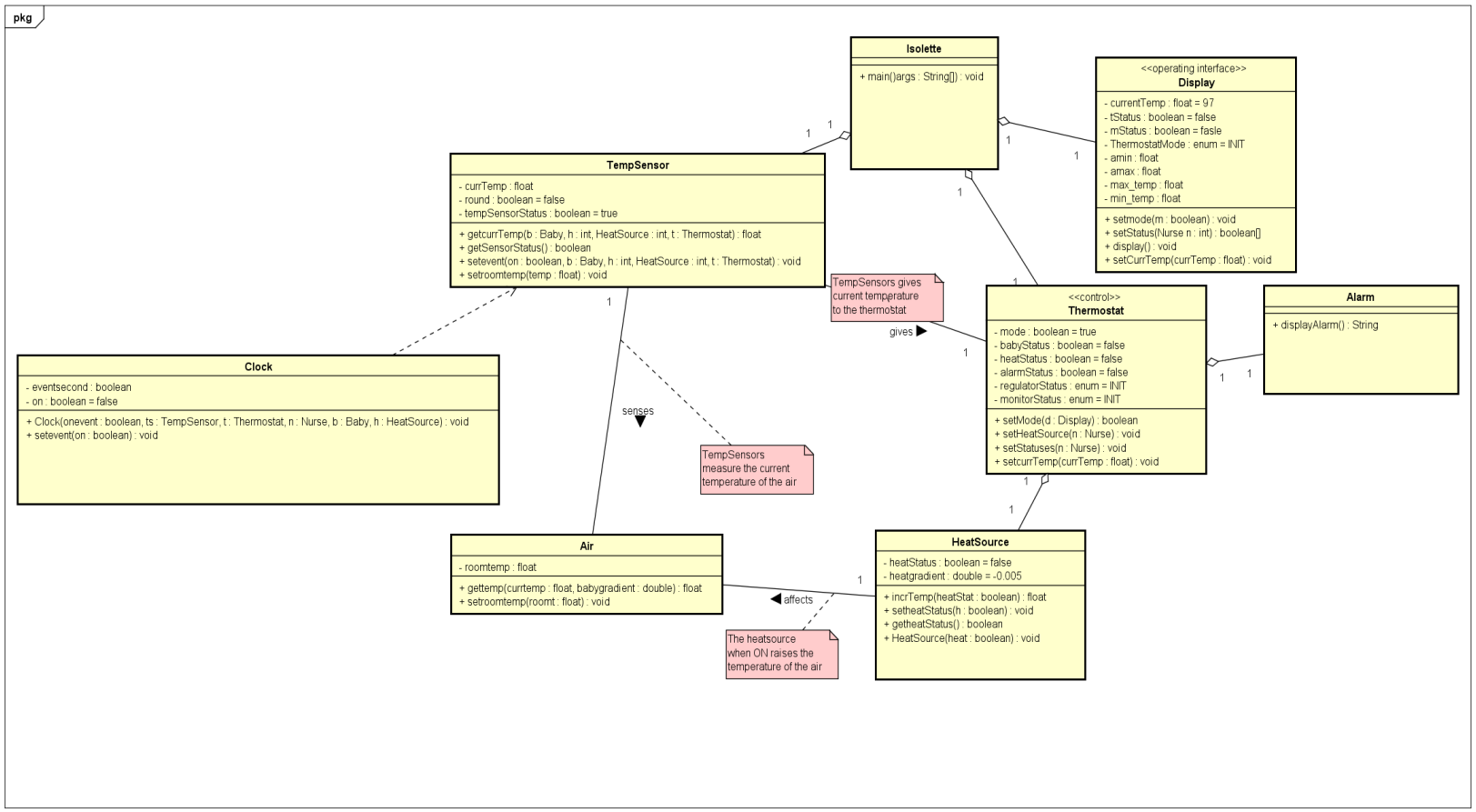
   It '*is a part (aggregation)'* of Thermostat class. There is a 1-1 relation between thermostat and heat souce. The heat source is *associated* with the Air class.

8. **TempSensor**
   TempSensor '*is a part (aggregation)'* of the isolette. For every isolette there is one TempSensor.

   The get, set methods for sensing the current temperature of the Air are used. The tempsensor calls the method of the Air class to sense the current temperature of the air and then pass the current temperature to the Thermostat class. The TempSenor senses the temperature of the Air continuously and sends it for further use to other class every 6 seconds i.e every round. The class is triggered every 6 seconds by the Clock.

9. **Isolette**
   The isolette is the main block from where all the other components are initiated. It has the main() and reads the file for the initial room temperature. All the components are a part of this system.

10. **Clock**
    Clock is the main synchronizing class of the system. The clock '*is a part (aggregation)'* of TempSenor. Every 6 seconds the Clock triggers the TempSensor to measure the current temperature. We perform multithreading in this class to call initiate the working of Regulator and Monitor interface (sub parts of Thermostat) simultaneously while the tempsensor is still measuring the current temperature.

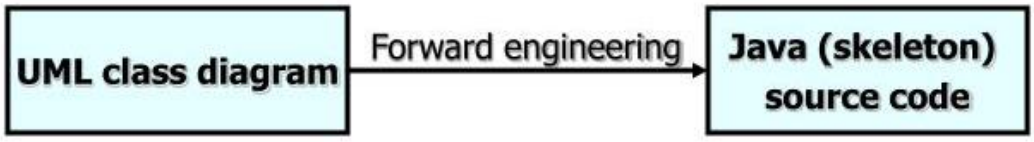


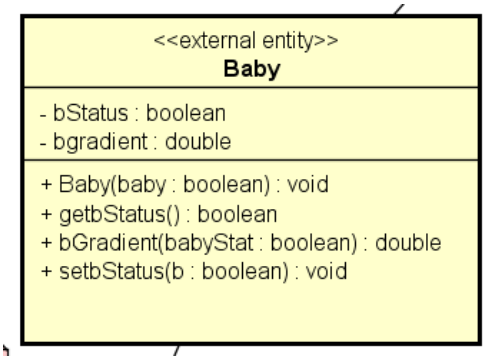

We have divided the components of the system in such a manner that the behavior of one component is different from other. Whenever a method is called from another class or class itself we have taken care that the necessary reactions are executed in a proper manner so that the other classes which are not associated in that interaction are not affected.
The states of the classes are maintained and synchronized.
*Design-by- Contract* is key methodology followed by us to develop the software of high quality. We categorize the creation, destruction and operations of the objects of a class properly while making sure that the execution of the sequence of operations on the classes in maintained.

## 3 IMPLEMENTATION

It provides the list of implemented classes, interface, and other classifiers [3]. For each implemented classifier provide its number of lines and size. The list of other files such as input and output files is to be included. A brief description on forward engineering the Astah UML designs to code should be provided.

The UML diagrams are not just for the visual design of the system but also for object oriented modelling/implementation of the system. UML tools like **[7]** help us to form and then do forward engineering of the diagram. The main motive of forward engineering is to produce the software implementation with the specification and design of the system. We did forward engineering from the developed Class diagrams as they are the main UML based design models.




```java
/**
 * Baby affects the temperature of the room
 */
public class Baby extends User {

    /**
     * Depicts if the baby is put in the isolete or not
     */
    private boolean bStatus ;
    private double bgradient ;


    /**
     * get the status of presence of baby form the thermostat.
     * Baby is put inside when the temperature of the isolette is
     * within the range of the desired ranges entered by the nurse
     */
    public Baby(){

    }
    public Baby(Boolean baby) {

    }

    public boolean getbStatus() {

    }


    public double bGradient(Boolean babyStat) {


    }
    public void setbStatus(Boolean b) {

    }
}
```

To show the association and interactions relationships we use role names, visibility parameters for each attribute as well as the methods of each class. We have kept attributes private to a class and some of them as package visibility. Methods are public to all the classes.
*In the associations(dependency) of unidirection as in from Clock class to TempSensor class we have atribute of type TempSensor in Clock class. Class TempSensor is global and its attributes are used by Class Clock. Clock uses TempSenor as a parameter in its method. Hence, TempSensor is instantiated as a local variable in Clock.*

Similarly for the bidirectional classes having interactions in form of feedback like in Display and NUrse we have attributes of both types in the other class.

We have used generalisations i.e inheritance to show the type of relationships in the class.

We have used definitions for the attributes and methods that are forward engineered for the description of the class's implementation.

The classes implemented by us are:

1. Baby
2. Nurse
3. Display
4. Alarm
5. HeatSource
6. TempSensor
7. Isolette
8. Thermostat
9. CLock
10. Air

We have used A User Class to classify the types of users (external entities) of the system into two types:

- Baby
- Nurse

There is an input file which has the value of the initial room temperature of the room.

- Inputtemp.txt
  You can change the initial temperature of the rooms for the purpose of experiments.
  In this manner we avoid hardcoding the initial temperature for the system.

## 4 EXPERIMENTS AND RESULTS

It provides outputs for the normal operation, configuration, and temperature maintenance use-cases [5,6].

The different experiments incorporate the different possibilities of execution of the system based on the implementation and this phase is an attempt to validate and verify the system function in the actual implementation.

The first one is a bad input case, where the nurse has input a bad desired minimum temperature which is below (or above) the range. The system gives a regulator status error.

```
<terminated> Isolette (1) [Java Application] C:\Program Files\Java\jdk-11.0.2\bin\javaw.exe
Enter intruction for the isolette(ON/OFF):
true
Enter desired minimum temperature:
96
Enter desired maximum temperature:
99
Enter alarm minimum temperature:
95
Enter alarm maximum temperature:
102
Thermostat is in INIT mode
Desired Minimum error,  default 97.0
Regulator Status: FAILURE
Monitor Status: NORMAL
ALARM
 Mode not safe for the Baby
Thermostat has failed FAILUREMODE
Regulator Failure has occured
```

The next one is a conflicting inputs case, where the individual inputs are valid, but the desired min and alarm min are not at least 1°F apart. This shall trigger a regulator status failure and stop the execution with an error message on the display.

```
<terminated> Isolette (1) [Java Application] C:\Program Files\Java\jdk-11.0.2\bin\javaw.exe
Enter intruction for the isolette(ON/OFF):
true
Enter desired minimum temperature:
98.5
Enter desired maximum temperature:
99
Enter alarm minimum temperature:
95
Enter alarm maximum temperature:
102
Thermostat is in INIT mode
Desired Maximum error, default 100.0
Regulator Status: FAILURE
Monitor Status: NORMAL
ALARM
 Mode not safe for the Baby
Thermostat has failed FAILUREMODE
Regulator Failure has occured
```

The next case will return with an error message under regulator status because the allowed value of the maximum desired temperature is 100°F, while the input value provided by the nurse is 101°F (EA-OI-7).

```
<terminated> Isolette (1) [Java Application] C:\Program Files\Java\jdk-11.0.2\bin\javaw.ex
Enter intruction for the isolette(ON/OFF):
true
Enter desired minimum temperature:
98
Enter desired maximum temperature:
101
Enter alarm minimum temperature:
95
Enter alarm maximum temperature:
103
Thermostat is in INIT mode
Desired Maximum error, default 100.0
Regulator Status: FAILURE
Monitor Status: NORMAL
ALARM
 Mode not safe for the Baby
Thermostat has failed FAILUREMODE
Regulator Failure has occured
```

This case will return with a monitor status error message as the alarm minimum temperature is 92°F, which is below the allowed range of [93, 97] (Appendix: EA-OI-3). The system will not allow the nurse to proceed and the nurse won't be able to put the baby in the isolette.

```
<terminated> Isolette (1) [Java Application] C:\Program Files\Java\jdk-11.0.2\bin\
Enter intruction for the isolette(ON/OFF):
true
Enter desired minimum temperature:
97
Enter desired maximum temperature:
99
Enter alarm minimum temperature:
92
Enter alarm maximum temperature:
102
Thermostat is in INIT mode
Alarm Minimum error, default 93.0
Regulator Status: NORMAL
Monitor Status: FAILURE
ALARM
 Mode not safe for the Baby
Thermostat has failed FAILUREMODE
Regulator Failure has occured
```

The next one is a similar case to the previous one, but the alarm minimum temperature entered is 97.5, which is greater than the allowed range [93, 97] (Appendix: EA-OI-5)

```
<terminated> Isolette (1) [Java Application] C:\Program Files\Java\jdk-1
Enter intruction for the isolette(ON/OFF):
true
Enter desired minimum temperature:
98
Enter desired maximum temperature:
99
Enter alarm minimum temperature:
97.5
Enter alarm maximum temperature:
102
Thermostat is in INIT mode
Alarm Minimum error, default 93.0
Regulator Status: NORMAL
Monitor Status: FAILURE
ALARM
 Mode not safe for the Baby
Thermostat has failed FAILUREMODE
Regulator Failure has occured
```

This is a case where the alarm maximum is 98.5°F, which is below the permissible range [99, 103] mentioned in the specifications document (EA-OI-9). This will give a monitor status failure and the nurse will not be allowed to put the baby inside the isolette.

```
<terminated> Isolette (1) [Java Application] C:\Program Files\Java\jdk-11.0.2\bin\javaw.e
Enter intruction for the isolette(ON/OFF):
true
Enter desired minimum temperature:
97
Enter desired maximum temperature:
99
Enter alarm minimum temperature:
94
Enter alarm maximum temperature:
98.5
Thermostat is in INIT mode
Alarm Maximum error, default 103.0
Regulator Status: NORMAL
Monitor Status: FAILURE
ALARM
 Mode not safe for the Baby
Thermostat has failed FAILUREMODE
Regulator Failure has occured
```

This case has both types of errors, a desired maximum error and an alarm maximum error. Both values are greater than their permissible values. As a result, the system gives two status errors: regulator status failure and monitor status failure while entering a failure mode. The nurse will not be allowed to perform normal execution in this case also because the inputs are not valid (EA-OI-9).

```
<terminated> Isolette (1) [Java Application] C:\Program Files\Java\jdk-
Enter intruction for the isolette(ON/OFF):
true
Enter desired minimum temperature:
98
Enter desired maximum temperature:
101
Enter alarm minimum temperature:
95
Enter alarm maximum temperature:
104
Thermostat is in INIT mode
Desired Maximum error, default 100.0
Alarm Maximum error, default 103.0
Regulator Status: FAILURE
Monitor Status: FAILURE
ALARM
 Mode not safe for the Baby
Thermostat has failed FAILUREMODE
Regulator Failure has occured
```

This is the first such test case where all inputs are permissible and has no input conflict, so the system allows a normal execution. In this test case, we can see that the temperature in the isolette is initially 98°F, which decreases below 98°F as the heat source is initially off. As 98°F is the desired minimum temperature, the heat source turns on as soon as the current temperature goes below it, and the temperature starts increasing. We witness the temperature increasing until the desired maximum, and as soon as it goes above the desired maximum value 99°F, the heat source is turned off and the temperature inside the isolette starts decreasing again. It is worth noticing that the increase in temperature during these rounds is greater in magnitude than the decrease, because the baby's breathing causes a slight warming of the temperature and for simulation purposes, we have added the operation as frequently as the temperature is measured given that the baby is present inside the isolette.

```
<terminated> Isolette (1) [Java Application] C:\Program Fi
Enter intruction for the isolette(ON/OFF):
true
Enter desired minimum temperature:
98
Enter desired maximum temperature:
99
Enter alarm minimum temperature:
95
Enter alarm maximum temperature:
101
Thermostat is in INIT mode
Regulator Status: NORMAL
Monitor Status: NORMAL
Thermostat is working NORMALMODE
Enter intruction for the isolette(ON/OFF):
Display Temperature is 98
Current Temperature is 98.0
 Put Baby inside
Display Temperature is 98
Current Temperature is 97.94
 Heat Source is turned ON
Display Temperature is 98
Current Temperature is 98.04
 Baby is still inside
Display Temperature is 98
Current Temperature is 98.14
 Baby is still inside
Display Temperature is 98
Current Temperature is 98.24
 Baby is still inside
Display Temperature is 98
Current Temperature is 98.34
 Baby is still inside
Display Temperature is 98
Current Temperature is 98.439995
 Baby is still inside
Display Temperature is 99
Current Temperature is 98.53999
 Baby is still inside
Display Temperature is 99
Current Temperature is 98.63999
 Baby is still inside
```

```
<terminated> Isolette (1) [Java Application] C:\Progra
Display Temperature is 98
Current Temperature is 98.04
 Baby is still inside
Display Temperature is 98
Current Temperature is 98.14
 Baby is still inside
Display Temperature is 98
Current Temperature is 98.24
 Baby is still inside
Display Temperature is 98
Current Temperature is 98.34
 Baby is still inside
Display Temperature is 98
Current Temperature is 98.439995
 Baby is still inside
Display Temperature is 99
Current Temperature is 98.53999
 Baby is still inside
Display Temperature is 99
Current Temperature is 98.63999
 Baby is still inside
Display Temperature is 99
Current Temperature is 98.73999
 Baby is still inside
Display Temperature is 99
Current Temperature is 98.83999
 Baby is still inside
Display Temperature is 99
Current Temperature is 98.93999
 Baby is still inside
Display Temperature is 99
Current Temperature is 99.039986
 Heat Source is turned OFF
Display Temperature is 99
Current Temperature is 98.97999
 Baby is still inside
Display Temperature is 99
Current Temperature is 98.91999
 Baby is still inside
false
Isolette is turned OFF
```

These two cases have failure modes owing to the difference in temperatures between the input values have not been maintained by at least 1°F as prescribed. In the test case on the left, the difference between the desired minimum and the desired maximum temperature is only 0.5°F. In the right-side case, the alarm minimum and desired minimum are only 0.5°F apart. These cases cause respective errors and terminate execution (EA-OI-8).

```
<terminated> Isolette (1) [Java Application] C:\Program Files\Java
Enter intruction for the isolette(ON/OFF):
true
Enter desired minimum temperature:
98
Enter desired maximum temperature:
98.5
Enter alarm minimum temperature:
94
Enter alarm maximum temperature:
102
Thermostat is in INIT mode
Desired Maximum error, default 100.0
Regulator Status: FAILURE
Monitor Status: NORMAL
ALARM
 Mode not safe for the Baby
Thermostat has failed FAILUREMODE
Regulator Failure has occured
```

```
<terminated> Isolette (1) [Java Application] C:\Program Files\Jav
Enter intruction for the isolette(ON/OFF):
true
Enter desired minimum temperature:
97
Enter desired maximum temperature:
99
Enter alarm minimum temperature:
96.5
Enter alarm maximum temperature:
102
Thermostat is in INIT mode
Alarm Minimum error, default 93.0
Regulator Status: NORMAL
Monitor Status: FAILURE
ALARM
 Mode not safe for the Baby
Thermostat has failed FAILUREMODE
Regulator Failure has occured
```

In this case, the difference between the desired maximum and the alarm maximum is less than 1°F, so the system will give a monitor status error and enter the failure mode (EA-OI-8).

```
<terminated> Isolette (1) [Java Application] C:\Program Files\Java
Enter intruction for the isolette(ON/OFF):
true
Enter desired minimum temperature:
97
Enter desired maximum temperature:
100
Enter alarm minimum temperature:
96
Enter alarm maximum temperature:
100.5
Thermostat is in INIT mode
Alarm Maximum error, default 103.0
Regulator Status: NORMAL
Monitor Status: FAILURE
ALARM
 Mode not safe for the Baby
Thermostat has failed FAILUREMODE
Regulator Failure has occured
```

This case is one where the nurse has input valid values and the initial temperature of the isolette starts at 75 F. Being below the desired range, the heat source is turned on and it is important to note that the isolette displays that it is unsafe to put the baby inside as the temperature has not yet reached the desired temperature range.

```
Isolette (1) [Java Application] C:\Program Files\Java\jdk-11.0.2\bin\javaw.exe (Apr 19, 201
Enter intruction for the isolette(ON/OFF):
true
Enter desired minimum temperature:
97
Enter desired maximum temperature:
99
Enter alarm minimum temperature:
95
Enter alarm maximum temperature:
102
Thermostat is in INIT mode
Regulator Status: NORMAL
Monitor Status: NORMAL
Thermostat is working NORMALMODE
 Heat Source is turned ON
Enter intruction for the isolette(ON/OFF):
Display Temperature is 75
Current Temperature is 75.0
ALARM
 Temperature still not safe for the Baby
 Heat Source is still ON
Display Temperature is 75
Current Temperature is 75.08
ALARM
 Temperature still not safe for the Baby
 Heat Source is still ON
Display Temperature is 75
Current Temperature is 75.16
ALARM
 Temperature still not safe for the Baby
 Heat Source is still ON
```

## 5 CONCLUSIONS

The conclusion should include lessons learned, recommendations, and experiences you have had in this course. This section should also include self-evaluation for each team member as well as the team.

**Lessons Learned:**

There have been many useful concepts within the process of Software Design that we have learned as we have worked on this project, and overall in this class. One of the main points of emphasis that we have learned through this course, and especially this project, is that paying attention to every single detail is essential in the Software Design stage of the software development life cycle. For example, in our Isolette project, it is of utmost importance to first read through all of the requirements in the Isolette document so that we can come to know of the specifications of the system. Even if a small detail is missed, it can completely change the whole intended design of the system. So that was one thing that we truly learned the importance of.

Next, we learned the methods of developing SRC diagrams. This was learned through the homework, and especially this project itself. We learned the ways in which developing the different components of the intended system can help us develop UML class diagrams next. So overall, that transition from SRC's to UML class diagrams was a key step in which we gained valuable experience throughout the duration of working on this project. These design principles are things which will definitely be to our advantage in our future endeavors.

**Recommendations:**

The only thing which we thought would be a good change to syllabus is that there could have been more lectures and time dedicated to discussing the process of going from SRC's to UML class diagrams. It might be beneficial to students if there is more emphasis on this, so that the process of transitioning from SRC to UML might come more simply in the class project. But other than that, we loved how everything else was in this class throughout the semester. We have learned many useful skills through this class which will help us in the future, and we thoroughly enjoyed the class.

**Evaluations:**

| Name | Effort (out of 5) | Participation (out of 5) | Comments |
|---|---|---|---|
| Riddhi Patel | 5 | 5 | Put full effort into the project and contributed to all parts of project |
| Snehit Mikkilineni | 5 | 5 | Contributed to every task in project and collaborated with team and put in effort |
| Shivam Shah | 5 | 5 | Worked on every part of the project with full effort and collaborated with team |

**Team Evaluation:**

Overall, as a team, we have accomplished all of the required tasks of the project. In the design phase of the project, we have implemented all the required SRC components, and we have thoroughly done the UML class diagrams appropriately. And within the implementation phase, we have fully implemented a software program, which mimics the behavior of the Isolette described in the document. And in doing all of these tasks, we have followed all of the requirements and specifications provided in the Isolette document and ensured that those requirements are met in our design and implementation of this system.

**References**

[1] R. Alur, (2015), Principles of Cyber-Physical Systems, MIT Press.
[2] G. Booch, et al., (2007), Object Oriented Analysis and Design (OOAD), 3rd Ed., Addison Wesley.
[3] Java Platform, Standard Edition (Java SE), (2019), https://www.oracle.com/java/technologies/java-se.html
[4] OMG 2012. "Unified Modeling Language version 2.5.1". https://www.omg.org/spec/UML/2.5.1/.
[5] Requirements Engineering Management Handbook, (2009), DOT/FAA/AR-08/32, Air Traffic Organization NextGen & Operations Planning Office of Research and Technology Development, https://www.faa.gov/aircraft/air_cert/design_approvals/air_software/media/AR-08-32.pdf.
[6] H.S. Sarjoughian, (2019), Software Design Course Project, CSE 564: Software Design (2019 Spring) 2019Spring-T-CSE564-32043 (Blackboard).
[7] http://astah.net/editions/uml-new

## A APPENDICES

All the definitions and assumptions have been satisfied directly from the handbook **[5]**

The following environmental assumptions are made:

• EA-OI-1: All temperatures will be entered and displayed in degrees Fahrenheit. Rationale: Minimize the complexity of this example. An actual system would probably support Celsius or perhaps both Fahrenheit and Celsius

• EA4-OI-2: All temperatures will be set and displayed by the operators in increments of 1°F. Rationale: Marketing studies have shown that customers prefer to set temperatures in 1°F increments. A resolution 1°F is enough to be consistent with the functional and performance requirements specified in the rest of the document.

• EA-OI-3: The Lower Alarm Temperature will always be ≥93°F. Rationale: Exposure to temperatures less than 93°F will result in hypothermia, which can lead to death within a few minutes for severely ill preterm infants.

• EA-OI-4: The Lower Alarm Temperature will always be less than or equal to the Lower Desired Temperature of -1°F. Rationale: If the Lower Alarm Temperature is greater than or equal to the Lower Desired Temperature, the Alarm could be activated while the Current Temperature is still in the Desired Temperature Range.

• EA-OI-5: The Lower Desired Temperature will always be ≥97°F. Rationale: Exposing the Infant to temperatures lower than 97°F may result in excessive heat loss and drop in heart rate secondary to metabolic acidosis.

• EA-OI-6: The Lower Desired Temperature will always be less than or equal to the Upper Desired Temperature of -1°F. Rationale: If the Lower Desired Temperature is greater than or equal to the Upper Desired Temperature, it is unclear if the Heat Source should be on or off. This may result in excessive cycling of the Heat Source.

• EA-OI-7: The Upper Desired Temperature will always be ≤100°F. Rationale: Exposing the Infant to temperatures greater than 100°F may result in an incorrect diagnosis of fever resulting in aggressive evaluation (blood culture and lumbar puncture) and treatment for infection.

• EA-OI-8: The Upper Alarm Temperature will always be greater than or equal to the Upper Desired Temperature of 1°F. Rationale: If the Upper Alarm Temperature is less than or equal to the Upper Desired Temperature, the Alarm could be activated while the Current Temperature is still in the Desired Temperature Range.

• EA-OI-9: The Upper Alarm Temperature will always be ≤103°F.

The high-level requirements for the Thermostat Function are as follows:

• REQ-TH-1: The Thermostat shall set the value of the Heat Control. Rationale: A primary function of the Thermostat is to turn the Heat Control on and off to maintain the Current Temperature in the Isolette within the Desired Temperature Range, which is required by SR-1.

• REQ-TH-2: The Thermostat Function shall set the value of the Regulator Status. Rationale: SR-1 requires the Thermostat to provide an independent regulator function. The status of this function is provided to the Operator Interface by the Thermostat. The Operator Interface will use the Regulator Status and the Monitor Status to report the overall status of the Thermostat, which is required by SR-1.

• REQ-TH-3: The Thermostat shall set the value of the Display Temperature. Rationale: The Current Temperature is displayed on the Operator Interface to provide the operators with an additional means to confirm the Isolette is maintaining the temperature correctly. This value is provided by the Thermostat to the Operator Interface as the Display Temperature.

• REQ-TH-4: The Thermostat shall set the value of the Alarm Control. Rationale: A primary Thermostat Function is to activate the Alarm if the Isolette is unable to maintain the Current Temperature within the Alarm Temperature Range, which is required by SR-2.

• REQ-TH-5: The Thermostat shall set the value of the Monitor Status. Rationale: SR-2 requires the Thermostat to provide an independent monitor function. The status of this function must be provided to the Operator Interface, which will use it and the status of the regulator function to report the overall status of the thermostat.

The high-level requirements for the Regulate Temperature Function are as follows:

• REQ-RT-1: The Regulate Temperature Function shall set the value of the Heat Control. Rationale: The primary function of the Regulate Temperature Function is to turn the Heat Control on and off to maintain the Current Temperature in the Isolette within the Desired Temperature Range, as required by SR-1.

• REQ-RT-2: The Regulate Temperature Function shall set the value of the Regulator Status. Rationale: The status of the Regulate Temperature Function is provided to the Operator Interface so it can use the status of the Regulate Temperature and Monitor Temperature Functions to report the overall status of the Thermostat, as required by SR-1.

• REQ-RT-3: The Regulate Temperature Function shall set the value of the Display Temperature. Rationale: The Current Temperature of the Isolette is displayed on the Operator Interface to provide the operators with an additional means to confirm that the Isolette is maintaining the temperature correctly. This value is provided by the Regulate Temperature Function to the Operator Interface as the Display Temperature.

The requirements for the Regulator Status controlled variable are as follows:

• REQ-MRI-1: If the Regulator Mode is INIT, the Regulator Status shall be set to Init

• REQ-MRI-2: If the Regulator Mode is NORMAL, the Regulator Status shall be set to On.

• REQ-MRI-3: If the Regulator Mode is FAILED, the Regulator Status shall be set to Failed. Latency: < Max Operator Response Time Tolerance: N/A

The requirements for the Display Temperature controlled variable are as follows:

• REQ-MRI-4: If the Regulator Mode is NORMAL, the Display Temperature shall be set to the value of the Current Temperature rounded to the nearest integer. Rationale: Displaying the rounded value of the Current Temperature provides the the most accurate display of the Current Temperature possible using an integer

display. When combined with the accuracy of the Temperature Sensor (EA-TS-2), the Display Temperature should be within 0.6°F of the actual value.

• REQ-MRI-5: If the Regulator Mode is not NORMAL, the value of the Display Temperature is UNSPECIFIED. Rationale: In modes other than NORMAL, the value of Display Temperature is not meaningful and should not be used. Latency: < Max Operator Response Time Tolerance: ±0.6°F

The requirements for the Regulator Interface Failure internal variable are as follows:

• REQ-MRI-6: If the Status attribute of the Lower Desired Temperature or the Upper Desired Temperature is Invalid, the Regulator Interface Failure shall be set to True.

• REQ-MRI-7: If the Status attribute of the Lower Desired Temperature and the Upper Desired Temperature is Valid, the Regulator Interface Failure shall be set to False. Rationale: The Regulator Interface Failure internal variable indicates if any errors have occurred in sensing the Operator Interface monitored variables needed by the Regulate Temperature Function. Note that its initial value on power-up will always be True since the Status of the Lower Desired Temperature and the Upper Desired Temperature are initially Invalid. The requirements for the Desired Range internal variable are as follows:

• REQ-MRI-8: If the Regulator Interface Failure is False, the Desired Range shall be set to the Desired Temperature Range.

• REQ-MRI-9: If the Regulator Interface Failure is True, the Desired Range is UNSPECIFIED. Rationale: The Desired Range is only meaningful when there is not a Regulator Interface Failure. If there is, its value should not be used, and it can be set to any value.

The requirements for the Heat Control controlled variable are as follows:

• REQ-MHS-1: If the Regulator Mode is INIT, the Heat Control shall be set to Off. Rationale: A regulator that is initializing cannot regulate the Current Temperature of the Isolette and the Heat Control should be turned off.

• REQ-MHS-2: If the Regulator Mode is NORMAL and the Current Temperature is less than the Lower Desired Temperature, the Heat Control shall be set to On.

• REQ-MHS-3: If the Regulator Mode is NORMAL and the Current Temperature is greater than the Upper Desired Temperature, the Heat Control shall be set to Off.

• REQ-MHS-4: If the Regulator Mode is NORMAL and the Current Temperature is greater than or equal to the Lower Desired Temperature and less than or equal to the Upper Desired Temperature, the value of the Heat Control shall not be changed. Rationale: When the Isolette is warming towards the Upper Desired Temperature, the Heat Source should be left on until the Upper Desired Temperature is reached. In a similar fashion, if the Isolette is cooling towards the Lower Desired Temperature, the Heat Source should be left off until the Lower Desired Temperature is reached.

• REQ-MHS-5: If the Regulator Mode is FAILED, the Heat Control shall be set to Off. Rationale: In failed mode, the regulator cannot regulate the Current Temperature of the Isolette and the Heat Control should be turned off. Latency: < Allowed Heat Source Latency Tolerance: N/A.

The high-level requirements for the Monitor Temperature Function are as follows:

• REQ-MT-1: The Monitor Temperature Function shall set the value of the Alarm Control. Rationale: The primary function of the Monitor Temperature Function is to raise an alarm if the Isolette is unable to maintain the Current Temperature within the Alarm Temperature Range, as required by safety requirement SR-2.

• REQ-MT-2: The Monitor Temperature Function shall set the value of the Monitor Status. Rationale: Safety requirement SR-2 requires the Thermostat to provide an independent monitor function. The status of this function must be provided to the Operator Interface, which will use it and the status of the Regulate Temperature Function to report the overall status of the Thermostat, as required by safety requirement SR-2.

The requirements for the Monitor Status controlled variable are as follows:

• REQ-MMI-1: If the Manage Monitor Interface mode is INIT, the Monitor Status shall be set to Init.

• REQ-MMI-2: If the Manage Monitor Interface mode is NORMAL, the Monitor Status shall be set to On.

• REQ-MMI-3: If the Manage Monitor Interface mode is FAILED, the Monitor Status shall be set to Failed. Latency: < Max Operator Response Time Tolerance: N/A

The requirements for Monitor Interface Failure internal variable are as follows:

• REQ-MMI-4: If the Status attribute of the Lower Alarm Temperature or the Upper Alarm Temperature is Invalid, the Monitor Interface Failure shall be set to True.

• REQ-MMI-5: If the Status attribute of the Lower Alarm Temperature and the Upper Alarm Temperature is Valid, the Monitor Interface Failure shall be set to False. Rationale: The Monitor Interface Failure internal variable indicates if any errors have occurred in sensing the Operator Interface monitored variables needed by the Manage Temperature Function. Note that its initial value on power-up will always be True since the Status attribute of the Lower Alarm Temperature and the Upper Alarm Temperature will initially be Invalid. The requirements for Alarm Range Internal variable are as follows:

• REQ-MMI-6: If the Monitor Interface Failure is False, the Alarm Range variable shall be set to the Desired Temperature Range.

• REQ-MMI-7: If the Monitor Interface Failure is True, the Alarm Range variable is UNSPECIFIED. Rationale: The Alarm Range variable is only meaningful when there is not a Monitor Interface Failure. If there is, its value should not use, and it can be set to any value.