

# Fuzzy Laboratory 3

## Applications implemented using components with FLETPN models

### 1. Laboratory Objectives

- Acquiring the concepts
  - a description of a first-order system with components,
  - implementing components with FLETPN models,
  - controlling a system with a P, PI algorithm.

developing and testing an application using FLETPN models.

### 2. Application description

We consider a room heated by a boiler. The hot water in the boiler must ensure a certain desired temperature in the room but this is also influenced by the outside temperature. The architecture of the system is shown in Figure 2.1. The room is modeled by a first-order system with multiple inputs and a single output.

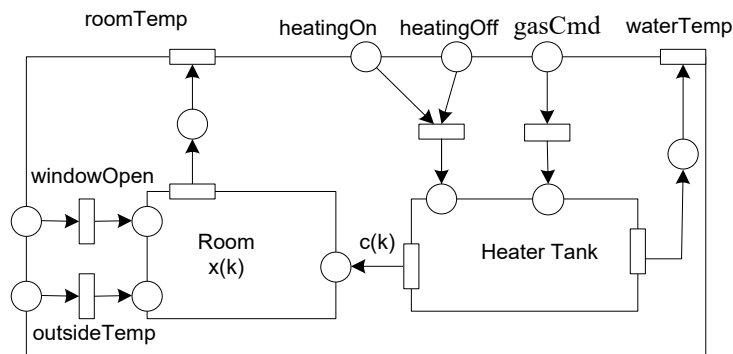


Figure 2.1 Plant component

The Room component models the room temperature and has the following input ports:

- *outsideTemp* - Indicates the outside temperature. This is considered a disturbance;
- *windowOpen* - informs whether the window is open or not. This is considered a disturbance;
- *c(k)* - heat given *HeaterTank*.

The Room component has a single exit port, *roomTemp*, which models the current system temperature.

Model for the room component:

$$x(k+1)=a*x(k)+b*c(k)+d1*v1(k)+d2*v1(k) \text{ where:}$$

$x(k)$  - room temperature;

$c(k)$  – heat given by *HeaterTank*;

$v_1$  – outside temperature;

$d_1$  – coefficient presenting cooling by opening the window;

$d_2$  – coefficient that models the cooling by walls;

$d_1 \cdot v_1(k)$  – the disturbance caused by opening the window;

$d_2 \cdot v_1(k)$  – disturbance caused by cooling by walls.

The HeaterTank component that models the boiler has the following ports:

- *heatingOn/heatingOff* – informs whether the heater is turned on or off. If it is turned on, then the water is recycled by radiators;
- *gasCmd* – is an input port variable that models how hard the gas is released.

The HeaterTank component has a  $c(k)$  output port, the HeaterTank heat, and the *waterTemp* port that models the current boiler temperature. When the outside temperature is very low, the boiler water temperature must be raised to warm the room.

The model for the HeaterTank component:

$$x_2(k+1) = a_2 \cdot x_2(k) + b_2 \cdot u(k) \text{ where:}$$

$x_2(k)$  is the temperature of the hot water in the *HeaterTank* and  $u(k)$  is the gas command.

### 3. The System Description

The control system consists of the component that models the room temperature (Room temperature Component- RTC) and the Heater Controller (Heather Temperature Controller - HTC) that controls the water temperature.

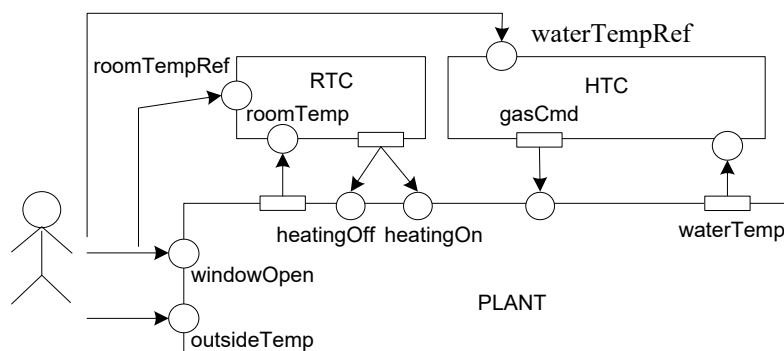


Figure 3.1 The System's Components

The RTC component is designed to keep the temperature as close as possible to the reference. The reference is given by a user through the *roomTempRef* input port. The *roomTemp* input port models the room temperature.

The HTC component is designed to heat the water to the desired temperature. Figure 3.1 shows the system's components with the input and output ports.

The mathematical model implements the logic:

IF  $(x(k) - \text{ref}) < \varepsilon$  THEN *heatingOn* is true

IF  $(x(k) - \text{ref}) > \varepsilon$  THEN *heatingOn* is false.

The *heatingOn* output port corresponds to the boiler startup, controls the water recycling and the room heating. The *heatingOff* port corresponds to stopping the boiler, stops the recycling.

The reference is given through the input port: *waterTempRef*. The input port for reading the current water temperature in the boiler is *waterTemp* and the continuous output port for the gas control is *gasCmd*. The HTC component implements a PI-type controller.

### 3.1 The Room Plant component model

The Room Plant Model is shown in figure 3.2.

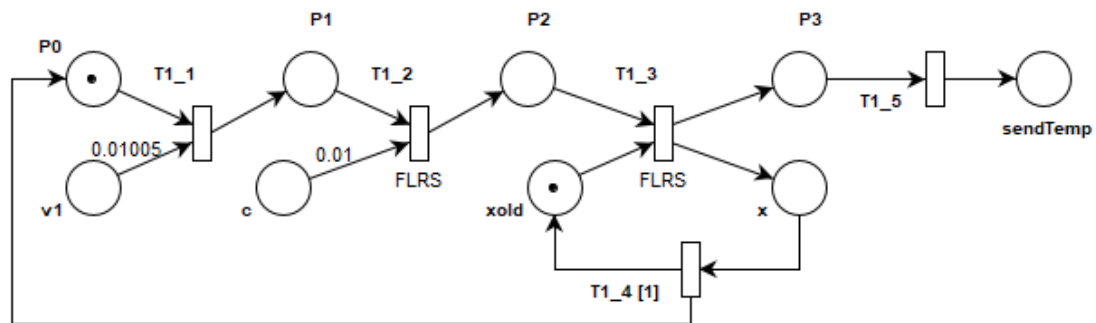


Figure 3.2 The Room Plant component model

The significance of ports and transitions is as follows:

v1 - is the input port for the outside temperature value;

c - is the input port for the heat sent by the Heater Tank component value;

T1\_1 - reads the outside Temperature multiplied by the coefficient  $(d1+d2)$  and saves it in P1;

T1\_2 adds the value of P1 ( $v1 * (d1+d2)$  if the window is open,  $v1 * d2$  if it is closed) to the heat sent by the heater tank multiplied by the coefficient b;

T1\_3 – computes the current room temperature  $x(k)$  by adding the value of P2 to the xold ( $x(k-1)$ ). Then stores the result in x and P3.

T1\_4 after a delay of 1 unit, gets  $x(k)$  and saves it in xold and in P0;

T1\_5 – Has an output channel (sendTemp) to send the  $x(k)$  to the RTC component model.

## 3.2 The Heater Tank component model

The Heater Tank component is shown in figure 3.3.

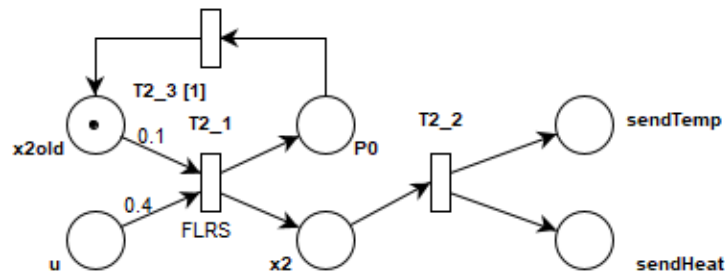


Figure 3.3 The Heater Tank component model

The significance of ports and transitions is as follows:

$u$  - is the input port for the command sent by the HTC component;

$T2\_1$  – computes the current water temperature  $x2$  by adding the value of the  $x2old$  ( $x2(k-1)$ ) multiplied by the coefficient  $a2$  to the command received in  $u$  multiplied by  $b2$  and saves the result in  $P0$  and  $x2$ ;

$T2\_2$  has two output channels to send  $x2$  through  $sendTemp$  to the RTC component and  $sendHeat$  to the Room Plant model;

$T2\_3$  after a delay of 1 unit, gets  $x2(k)$  from  $P0$  and saves it in  $x2old$ ;

## The RTC component model

The RTC component implements a bipolar controller. Figure 3.4 shows the Petri net model of the RTC component. The net is an FLETPN type, the transition  $T1$  has a FLRS differential type table, and the  $T3$  transition is attached to a FLRS comparator table.

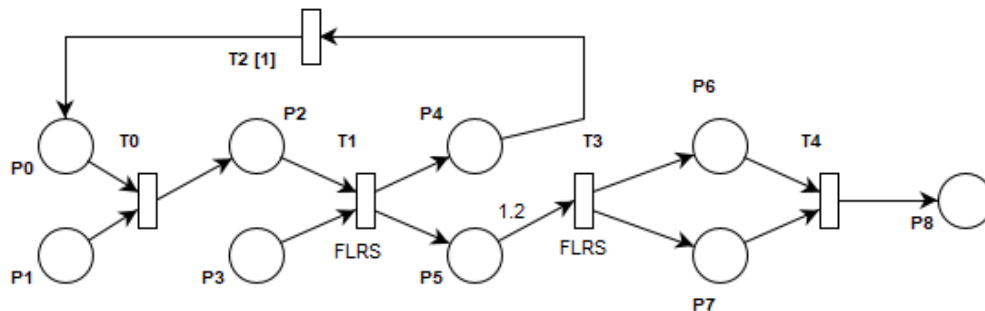


Figure 3.4 The RTC component model

The significance of ports and transitions is as follows:

P1 - is the input port for the *roomTempRef* value;

P3 - is the input port for the current *roomTemp* value;

T0 - reads the reference value and stores it in P2;

T1 - is a differential and calculates the *roomTempRef* – *roomTemp* error and stores the result in P4 and P5;

T2 - after a delay of 1 unit, the token in P4 is saved in P0;

T3 - Put a token in the place P6 if the calculated error is NL and for the PL in the place P7.

T4- Has an output channel P8 that is connected to the Heater Tank plant, if the token in P6 = NL (-1), then a command is sent to the heater tank to **stop** until the temperature reaches *roomTempRef* +  $\epsilon$  value, and if the token in P7 =PL (+1), a command is sent to the heater tank to **start**. It will turn on again when *roomTemp* drops below *roomTempRef* -  $\epsilon$ . The value of the variable  $\epsilon$  is the value corresponding to the weight of the weights at the place P5 at to the transition T3.

## 3.2 HTC component model

Figure 3.5 shows the Petri net that models the HTC component. The HTC component is built on the P model.

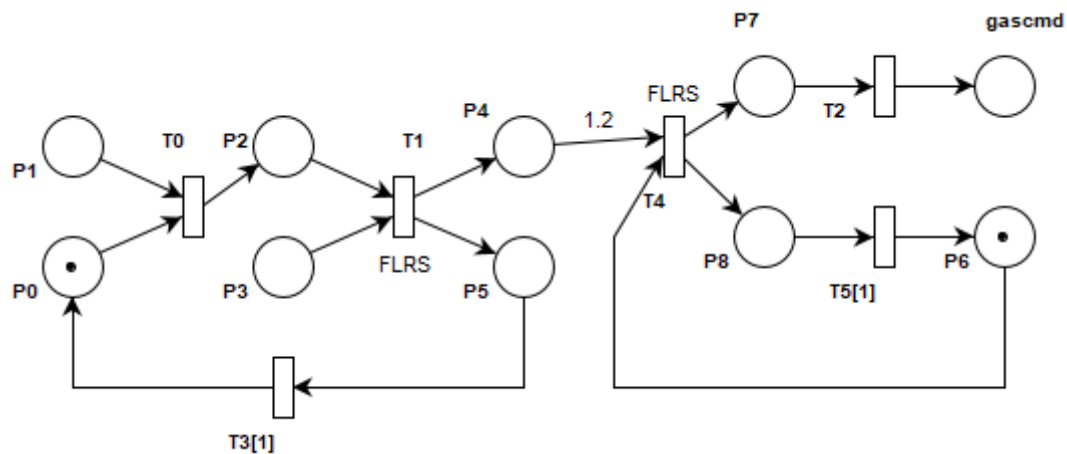


Figure 3.5 HTC component model

The significance of ports and transitions is as follows:

P1 - reads the temperature reference *waterTempRef*;

T0 - stores the reference at place P2;

P3 - reads the water temperature of the heater tank *waterTemp*;

T1 - calculates the error  $e(k) = waterTempRef(k) - waterTemp(k)$  and stores it in P4 and P5;

T3 - after a delay of 1 u.t. (time unit) the token in P5 is stored in P0;

T4 - computes the current value of the command  $u(k) = u(k-1) + \Delta u(k)$  by summing the two values stored in P7 and P8;

T2 - a continuous output port (gascmd) for the gas control and sends it to the heater tank plant componenet;

T5 –update command with a delay of 1 u.t. the command value towards the boiler,  $u(k) = u(k-1)$  and indicates the restart of a new calculation of the command.

## 4. Implementation in Java

The implementation of this system can be found in [https://github.com/dahliajanabi/All\\_Petri\\_Nets\\_Framework](https://github.com/dahliajanabi/All_Petri_Nets_Framework) DCS\_FuzzyLab/RoomHeatingSystem. First create a directory on your hard drive for the PetriInputData, then run the WheatherInput class, it has three scenarios: winterDay, winterMorning, and extremeEvening. Then run the RoomPlant and HeaterTankPlant first, then run the controllers (RTC and HTC) after. Click on show graph button on the RoomPlant to see the output. The code for the WheatherInput, RoomPlant, HeaterTankPlant, RTC, and HTC can also be found in code listings 1-5 subsequently.

### Code Listing 1: WheatherInput.java

```
package DCS_FuzzyLab.RoomHeatingSystem;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.file.Files;

//For testing the PI controller alone
public class WheatherInput {
    public static void main(String[] args) throws InterruptedException, IOException {
        File plant = new File("D:\\PetriInputData\\OutsideTemp.txt");
        Files.deleteIfExists(plant.toPath());

        File RTC = new File("D:\\PetriInputData\\RTC.txt");
        Files.deleteIfExists(RTC.toPath());

        File HTC = new File("D:\\PetriInputData\\HTC.txt");
        Files.deleteIfExists(HTC.toPath());

        // Scenarios:
        float winterDay[] = new float[] { -12.5F / 25, -15.0F / 25, -17.0F / 25, -
20.0F / 25, -21.0F / 25, -19.0F / 25,
-17.0F / 25, -15.0F / 25, -12.0F / 25, -8.0F / 25, -7.0F / 25,
-5.0F / 25, -4.0F / 25, -3.5F / 25,
-5.0F / 25, -4.0F / 25, -5.0F / 25, -6.0F / 25, -7.5F / 25, -
8.5F / 25, -9.0F / 25, -11.0F / 25,
-11.5F / 25, -12.0F / 25, -12.0F / 25 };
    }
```

```

        float winterMorning[] = new float[] { -19.0F / 25, -18.5F / 25, -18.0F / 25,
-17.5F / 25, -17.0F / 25,
                                -16.5F / 25, -16.0F / 25, -15.5F / 25, -15.0F / 25, -14.5F /
25, -14.0F / 25, -13.5F / 25, -13.0F / 25,
                                -12.5F / 25, -12.0F / 25 };

        float extermeEvening[] = new float[] { -5.0F / 25, -6.0F / 25, -8.0F / 25, -
10.0F / 25, -12.0F / 25,
                                -13.0F / 25, -14.0F / 25, -16.0F / 25, -18.0F / 25, -22.0F /
25, -27.0F / 25 };
        // -----

        float roomReference = 24 / 25F;
        float waterReference = 48 / 70F;

        FileWriter fwPlant = new FileWriter(plant.getPath());
        FileWriter fwRTC = new FileWriter(RTC.getPath());
        FileWriter fwHTC = new FileWriter(HTC.getPath());

        for (int i = 0; i < winterDay.length; i++) {
            fwPlant.write("v1:" + winterDay[i] + "F\n");
            fwRTC.write("P1:" + roomReference + "F\n");
            fwHTC.write("P1:" + waterReference + "F\n");
        }

        fwPlant.close();
        fwRTC.close();
        fwHTC.close();
        System.out.println("Done!");
    }
}

```

## Code Listing 2: RoomPlant.java

```

package DCS_FuzzyLab.RoomHeatingSystem;

import java.io.FileNotFoundException;
import java.util.ArrayList;
import Components.Activation;
import Components.Condition;
import Components.GuardMapping;
import Components.PetriNet;
import Components.PetriNetWindow;
import Components.PetriTransition;
import DataObjects.DataFuzzy;
import DataObjects.DataTransfer;
import DataOnly.FLRS;
import DataOnly.FV;
import DataOnly.Fuzzy;
import DataOnly.PlaceNameWithWeight;
import DataOnly.TransferOperation;
import Enumerations.FZ;
import Enumerations.LogicConnector;
import Enumerations.TransitionCondition;
import Enumerations.TransitionOperation;

/*
    The Equations are:
    *      xNew = a * x + b * c + d1 * v1 + d2 * v2;
    *      The constants are:
    *      a = 1.0f;

```

```

*           b = 0.01f;
*           d1 = 0.01f;
*           d2 = 0.00055f;
*/
public class RoomPlant {
    public static void main(String[] args) throws FileNotFoundException {
        PetriNet pn = new PetriNet();
        pn.PetriNetName = "Room Plant";
        pn.NetworkPort = 1080;

        pn.SetInputFile("D:\\PetriInputData\\OutsideTemp.txt");

        FLRS reader = new FLRS(new FV(FZ.NL), new FV(FZ.NM), new FV(FZ.ZR), new
FV(FZ.PM), new FV(FZ.PL),
                                new FV(FZ.NL), new FV(FZ.NM), new FV(FZ.ZR), new FV(FZ.PM),
new FV(FZ.PL),
                                new FV(FZ.NL), new FV(FZ.NM), new FV(FZ.ZR), new FV(FZ.PM),
new FV(FZ.PL),
                                new FV(FZ.NL), new FV(FZ.NM), new FV(FZ.ZR), new FV(FZ.PM),
new FV(FZ.PL));

        FLRS adder = new FLRS(new FV(FZ.NL), new FV(FZ.NL), new FV(FZ.NL), new
FV(FZ.NM), new FV(FZ.ZR),
                                new FV(FZ.NL), new FV(FZ.NL), new FV(FZ.NM), new FV(FZ.ZR),
new FV(FZ.PM),
                                new FV(FZ.NL), new FV(FZ.NM), new FV(FZ.ZR), new FV(FZ.PM),
new FV(FZ.PL),
                                new FV(FZ.NM), new FV(FZ.ZR), new FV(FZ.PM), new FV(FZ.PL),
new FV(FZ.PL),
                                new FV(FZ.ZR), new FV(FZ.PM), new FV(FZ.PL), new FV(FZ.PL),
new FV(FZ.PL));

        FLRS doubleChannelAdder = new FLRS(new FV(FZ.NL, FZ.NL), new FV(FZ.NL,
FZ.NL), new FV(FZ.NL, FZ.NL), new FV(FZ.NM, FZ.NM), new FV(FZ.ZR, FZ.ZR),
                                new FV(FZ.NL, FZ.NL), new FV(FZ.NL, FZ.NL), new FV(FZ.NM,
FZ.NM), new FV(FZ.ZR, FZ.ZR), new FV(FZ.PM, FZ.PM),
                                new FV(FZ.NL, FZ.NL), new FV(FZ.NM, FZ.NM), new FV(FZ.ZR,
FZ.ZR), new FV(FZ.PM, FZ.PM), new FV(FZ.PL, FZ.PL),
                                new FV(FZ.NM, FZ.NM), new FV(FZ.ZR, FZ.ZR), new FV(FZ.PM,
FZ.PM), new FV(FZ.PL, FZ.PL), new FV(FZ.PL, FZ.PL),
                                new FV(FZ.ZR, FZ.ZR), new FV(FZ.PM, FZ.PM), new FV(FZ.PL,
FZ.PL), new FV(FZ.PL, FZ.PL), new FV(FZ.PL, FZ.PL));

        FLRS OneXTwoDefaultTable = new FLRS(new FV(FZ.NL, FZ.NL), new FV(FZ.NM, FZ.NM),
new FV(FZ.ZR, FZ.ZR), new FV(FZ.PM, FZ.PM), new FV(FZ.PL, FZ.PL));

        reader.Print();
        adder.Print();
        doubleChannelAdder.Print();
        OneXTwoDefaultTable.Print();

        DataFuzzy xold = new DataFuzzy();
        xold.SetName("xold");
        xold.SetValue(new Fuzzy(0.92F)); // start temp 23 and temp range -25 - 25
        pn.PlaceList.add(xold);

        DataFuzzy x = new DataFuzzy();
        x.SetName("x");

```

```

pn.PlaceList.add(x);

DataFuzzy c = new DataFuzzy(); // from Heater Tank Plant
c.SetName("c");
pn.PlaceList.add(c);

DataFuzzy v1 = new DataFuzzy(); // outside temp
v1.SetName("v1");
pn.PlaceList.add(v1);

DataFuzzy p0 = new DataFuzzy();
p0.SetName("p0");
p0.SetValue(new Fuzzy(0.0F));
pn.PlaceList.add(p0);

DataFuzzy p1 = new DataFuzzy();
p1.SetName("p1");
pn.PlaceList.add(p1);

DataFuzzy p2 = new DataFuzzy();
p2.SetName("p2");
pn.PlaceList.add(p2);

DataFuzzy p3 = new DataFuzzy();
p3.SetName("p3");
pn.PlaceList.add(p3);

DataFuzzy p4 = new DataFuzzy();
p4.SetName("p4");
pn.PlaceList.add(p4);

DataTransfer sendTemp = new DataTransfer();
sendTemp.SetName("sendTemp");
sendTemp.Value = new TransferOperation("localhost", "1082", "P3"); // to RTC
pn.PlaceList.add(sendTemp);

// t_11 -----
PetriTransition t_11 = new PetriTransition(pn);
t_11.TransitionName = "t_11";
t_11.InputPlaceName.add("p0");
t_11.InputPlaceName.add("v1");

Condition T_11Ct1 = new Condition(t_11, "v1", TransitionCondition.NotNull);
Condition T_11Ct2 = new Condition(t_11, "p0", TransitionCondition.NotNull);
T_11Ct1.SetNextCondition(LogicConnector.AND, T_11Ct2);

GuardMapping grdt_11 = new GuardMapping();
grdt_11.condition = T_11Ct1;

ArrayList<PlaceNameWithWeight> input1 = new ArrayList<>();
input1.add(new PlaceNameWithWeight("p0", 1F));
input1.add(new PlaceNameWithWeight("v1", 0.01005F)); // window disturbance +
Walls v1(d1+d2)

ArrayList<String> output1 = new ArrayList<>();
output1.add("p1");

grdt_11.Activations.add(new Activation(t_11, reader, input1,
TransitionOperation.FLRS, output1));

t_11.GuardMappingList.add(grdt_11);

```

```

t_11.Delay = 0;
pn.Transitions.add(t_11);

// t_12 -----
PetriTransition t_12 = new PetriTransition(pn);
t_12.TransitionName = "t_12";
t_12.InputPlaceName.add("p1");
t_12.InputPlaceName.add("c");

Condition T_12Ct1 = new Condition(t_12, "c", TransitionCondition.NotNull);
Condition T_12Ct2 = new Condition(t_12, "p1", TransitionCondition.NotNull);
T_12Ct1.SetNextCondition(LogicConnector.AND, T_12Ct2);

GuardMapping grdt_12 = new GuardMapping();
grdt_12.condition = T_12Ct1;

ArrayList<PlaceNameWithWeight> input2 = new ArrayList<>();
input2.add(new PlaceNameWithWeight("p1", 1F));
input2.add(new PlaceNameWithWeight("c", 0.01F));

ArrayList<String> output2 = new ArrayList<>();
output2.add("p2");

grdt_12.Activations.add(new Activation(t_12, adder, input2,
TransitionOperation.FLRS, output2));

t_12.GuardMappingList.add(grdt_12);

t_12.Delay = 0;
pn.Transitions.add(t_12);

// t_13 -----
PetriTransition t_13 = new PetriTransition(pn);
t_13.TransitionName = "t_13";
t_13.InputPlaceName.add("p2");
t_13.InputPlaceName.add("xold");

Condition T_13Ct1 = new Condition(t_13, "xold", TransitionCondition.NotNull);
Condition T_13Ct2 = new Condition(t_13, "p2", TransitionCondition.NotNull);
T_13Ct1.SetNextCondition(LogicConnector.AND, T_13Ct2);

GuardMapping grdt_13 = new GuardMapping();
grdt_13.condition = T_13Ct1;

ArrayList<PlaceNameWithWeight> input3 = new ArrayList<>();
input3.add(new PlaceNameWithWeight("p2", 1F));
input3.add(new PlaceNameWithWeight("xold", 1F));

ArrayList<String> output3 = new ArrayList<>();
output3.add("x");
output3.add("p3");

grdt_13.Activations.add(new Activation(t_13, doubleChannelAdder, input3,
TransitionOperation.FLRS, output3));

t_13.GuardMappingList.add(grdt_13);

t_13.Delay = 0;
pn.Transitions.add(t_13);

```

```

// t_14 -----
PetriTransition t_14 = new PetriTransition(pn);
t_14.TransitionName = "t_14";
t_14.InputPlaceName.add("x");

Condition T_14Ct1 = new Condition(t_14, "x", TransitionCondition.NotNull);

GuardMapping grdt_14 = new GuardMapping();
grdt_14.condition = T_14Ct1;

ArrayList<PlaceNameWithWeight> input4 = new ArrayList<>();
input4.add(new PlaceNameWithWeight("x", 1F));

ArrayList<String> output4 = new ArrayList<>();
output4.add("xold");
output4.add("p0");

grdt_14.Activations.add(new Activation(t_14, OneXTwoDefaultTable, input4,
TransitionOperation.FLRS, output4));

t_14.GuardMappingList.add(grdt_14);

t_14.Delay = 1;
pn.Transitions.add(t_14);

// t_15 -----
PetriTransition t_15 = new PetriTransition(pn);
t_15.TransitionName = "t_15";
t_15.InputPlaceName.add("p3");

Condition T_15Ct1 = new Condition(t_14, "p3", TransitionCondition.NotNull);

GuardMapping grdt_15 = new GuardMapping();
grdt_15.condition = T_15Ct1;

grdt_15.Activations.add(new Activation(t_15, "p3",
TransitionOperation.SendOverNetwork, "sendTemp"));

t_15.GuardMappingList.add(grdt_15);

t_15.Delay = 0;
pn.Transitions.add(t_15);

//-----

System.out.println("Room Plant started \n -----");
pn.Delay = 500;
//pn.PrintingSpeed=0;
pn.ShowLogInWindow=true;
// pn.Start();

PetriNetWindow frame = new PetriNetWindow(false);
frame.petriNet = pn;
frame.setVisible(true);
}
}

```

### Code Listing 3: HeaterTankPlant.java

```
package DCS_FuzzyLab.RoomHeatingSystem;

import java.io.FileNotFoundException;
import java.util.ArrayList;
import Components.Activation;
import Components.Condition;
import Components.GuardMapping;
import Components.PetriNet;
import Components.PetriNetWindow;
import Components.PetriTransition;
import DataObjects.DataFuzzy;
import DataObjects.DataTransfer;
import DataOnly.FLRS;
import DataOnly.FV;
import DataOnly.Fuzzy;
import DataOnly.PlaceNameWithWeight;
import DataOnly.TransferOperation;
import Enumerations.FZ;
import Enumerations.LogicConnector;
import Enumerations.TransitionCondition;
import Enumerations.TransitionOperation;

/*
    The Equations are:
    *       $x2_{New} = a2 * x2 + b2 * u;$ 
    *      The constants are:
    *           $a2 = 0.1f;$ 
    *           $b2 = 0.4f;$ 
    */
public class HeaterTankPlant {
    public static void main(String[] args) throws FileNotFoundException {
        PetriNet pn = new PetriNet();
        pn.PetriNetName = "Heater Tank Plant";
        pn.NetworkPort = 1081;

        FLRS doubleChannelAdder = new FLRS(new FV(FZ.NL, FZ.NL), new FV(FZ.NL,
FZ.NL), new FV(FZ.NL, FZ.NL), new FV(FZ.NM, FZ.NM), new FV(FZ.ZR, FZ.ZR),
new FV(FZ.NL,
FZ.NL), new FV(FZ.NL, FZ.NL), new FV(FZ.NM, FZ.NM), new FV(FZ.ZR, FZ.ZR), new FV(FZ.PM,
FZ.PM),
new FV(FZ.NL,
FZ.NL), new FV(FZ.NM, FZ.NM), new FV(FZ.ZR, FZ.ZR), new FV(FZ.PM, FZ.PM), new FV(FZ.PL,
FZ.PL),
new FV(FZ.NM,
FZ.NM), new FV(FZ.ZR, FZ.ZR), new FV(FZ.PM, FZ.PM), new FV(FZ.PL, FZ.PL), new FV(FZ.PL,
FZ.PL),
new FV(FZ.ZR,
FZ.ZR), new FV(FZ.PM, FZ.PM), new FV(FZ.PL, FZ.PL), new FV(FZ.PL, FZ.PL), new FV(FZ.PL,
FZ.PL));

        FLRS OneXOneDefaultTable = new FLRS(new FV(FZ.NL), new FV(FZ.NM), new
FV(FZ.ZR), new FV(FZ.PM), new FV(FZ.PL));

        doubleChannelAdder.Print();
        OneXOneDefaultTable.Print();

        DataFuzzy x2old = new DataFuzzy();
        x2old.SetName("x2old");
        x2old.SetValue(new Fuzzy(0.3F)); // start temp 23 and temp range max 75
        pn.PlaceList.add(x2old);
    }
}
```

HTC

```
DataFuzzy x2 = new DataFuzzy();
x2.SetName("x2");
pn.PlaceList.add(x2);

DataFuzzy p0 = new DataFuzzy();
p0.SetName("p0");
pn.PlaceList.add(p0);

DataFuzzy u = new DataFuzzy(); // from Heater Tank Plant
u.SetName("u");
pn.PlaceList.add(u);

DataFuzzy runningState = new DataFuzzy(); // from Heater Tank Plant
runningState.SetName("Running State");
pn.PlaceList.add(runningState);

DataTransfer sendHeat = new DataTransfer();
sendHeat.SetName("sendHeat");
sendHeat.Value = new TransferOperation("localhost", "1080", "c"); // to Room
pn.PlaceList.add(sendHeat);

DataTransfer sendWaterTemp = new DataTransfer();
sendWaterTemp.SetName("sendTemp");
sendWaterTemp.Value = new TransferOperation("localhost", "1083", "P3"); // to

pn.PlaceList.add(sendWaterTemp);

// t_21 -----
PetriTransition t_21 = new PetriTransition(pn);
t_21.TransitionName = "t_21";
t_21.InputPlaceName.add("x2old");
t_21.InputPlaceName.add("u");

Condition T_12Ct1 = new Condition(t_21, "x2old",
TransitionCondition.NotNull);
Condition T_12Ct2 = new Condition(t_21, "u", TransitionCondition.NotNull);
T_12Ct1.SetNextCondition(LogicConnector.AND, T_12Ct2);

GuardMapping grdt_12 = new GuardMapping();
grdt_12.condition = T_12Ct1;

ArrayList<PlaceNameWithWeight> input1 = new ArrayList<>();
input1.add(new PlaceNameWithWeight("x2old", 0.1F));
input1.add(new PlaceNameWithWeight("u", 0.4F));

ArrayList<String> output1 = new ArrayList<>();
output1.add("p0");
output1.add("x2");

grdt_12.Activations.add(new Activation(t_21, doubleChannelAdder, input1,
TransitionOperation.FLRS, output1));

t_21.GuardMappingList.add(grdt_12);

t_21.Delay = 0;
pn.Transitions.add(t_21);

// t_22 -----
PetriTransition t_22 = new PetriTransition(pn);
t_22.TransitionName = "t_22";
t_22.InputPlaceName.add("x2");
```

```

        Condition T_22Ct1 = new Condition(t_22, "x2", TransitionCondition.NotNull);

        GuardMapping grdt_22 = new GuardMapping();
        grdt_22.condition = T_22Ct1;

        grdt_22.Activations.add(new Activation(t_22, "x2",
TransitionOperation.SendOverNetwork, "sendHeat"));
        grdt_22.Activations.add(new Activation(t_22, "x2",
TransitionOperation.SendOverNetwork, "sendTemp"));

        t_22.GuardMappingList.add(grdt_22);

        t_22.Delay = 1;
        pn.Transitions.add(t_22);

        // t_23 -----
        PetriTransition t_23 = new PetriTransition(pn);
        t_23.TransitionName = "t_23";
        t_23.InputPlaceName.add("p0");

        Condition T_23Ct1 = new Condition(t_23, "p0", TransitionCondition.NotNull);

        GuardMapping grdt_23 = new GuardMapping();
        grdt_23.condition = T_23Ct1;

        ArrayList<PlaceNameWithWeight> input2 = new ArrayList<>();
        input2.add(new PlaceNameWithWeight("p0", 0.1F));

        ArrayList<String> output2 = new ArrayList<>();
        output2.add("x2old");

        grdt_23.Activations.add(new Activation(t_23, OneXOneDefaultTable, input2,
TransitionOperation.FLRS, output2));

        t_23.GuardMappingList.add(grdt_23);

        t_23.Delay = 1;
        pn.Transitions.add(t_23);

        // -----

        System.out.println("Heater Tank started \n -----");
        pn.Delay = 500;
        //pn.PrintingSpeed=0;
        pn.ShowLogInWindow=true;
        // pn.Start();

        PetriNetWindow frame = new PetriNetWindow(false);
        frame.petriNet = pn;
        frame.setVisible(true);
    }
}

```

#### Code Listing 4: RTC

```

package DCS_FuzzyLab.RoomHeatingSystem;

import java.io.FileNotFoundException;
import java.util.ArrayList;

```

```

import Components.Activation;
import Components.Condition;
import Components.GuardMapping;
import Components.PetriNet;
import Components.PetriNetWindow;
import Components.PetriTransition;
import DataObjects.DataFuzzy;
import DataObjects.DataNetworkCommand;
import DataObjects.DataTransfer;
import DataOnly.FLRS;
import DataOnly.FV;
import DataOnly.Fuzzy;
import DataOnly.NetworkCommand;
import DataOnly.PlaceNameWithWeight;
import DataOnly.TransferOperation;
import Enumerations.FZ;
import Enumerations.LogicConnector;
import Enumerations.NetworkCommands;
import Enumerations.TransitionCondition;
import Enumerations.TransitionOperation;

public class RTC {
    public static void main (String[]args) throws FileNotFoundException {
        PetriNet pn = new PetriNet();
        pn.PetriNetName = "RTC";
        pn.NetworkPort = 1082;

        pn.SetInputFile("D:\\PetriInputData\\RTC.txt");

        FLRS reader = new FLRS(new FV(FZ.NL), new FV(FZ.NM), new FV(FZ.ZR), new
FV(FZ.PM), new FV(FZ.PL),
                                new FV(FZ.NL), new FV(FZ.NM), new
FV(FZ.ZR), new FV(FZ.PM), new FV(FZ.PL),
                                new FV(FZ.NL), new FV(FZ.NM), new
FV(FZ.ZR), new FV(FZ.PM), new FV(FZ.PL),
                                new FV(FZ.NL), new FV(FZ.NM), new
FV(FZ.ZR), new FV(FZ.PM), new FV(FZ.PL));

        FLRS doubleChannelDifferentiator = new FLRS(new FV(FZ.ZR, FZ.ZR), new
FV(FZ.NM, FZ.NM), new FV(FZ.NL, FZ.NL), new FV(FZ.NL, FZ.NL),new FV(FZ.NL, FZ.NL),
                                                                new
FV(FZ.PM, FZ.PM), new FV(FZ.ZR, FZ.ZR), new FV(FZ.NM, FZ.NM), new FV(FZ.NL,
FZ.NL),new FV(FZ.NL, FZ.NL),
                                                                new
FV(FZ.PL, FZ.PL), new FV(FZ.PM, FZ.PM), new FV(FZ.ZR, FZ.ZR), new FV(FZ.NM,
FZ.NM),new FV(FZ.NL, FZ.NL),
                                                                new
FV(FZ.PL, FZ.PL), new FV(FZ.PL, FZ.PL), new FV(FZ.PM, FZ.PM), new FV(FZ.ZR,
FZ.ZR),new FV(FZ.NM, FZ.NM),
                                                                new
FV(FZ.PL, FZ.PL), new FV(FZ.PL, FZ.PL), new FV(FZ.PL, FZ.PL), new FV(FZ.PM,
FZ.PM),new FV(FZ.ZR, FZ.ZR));

```

```

    FLRS OneXOneDefaultTable = new FLRS(new FV(FZ.NL), new FV(FZ.NM), new
FV(FZ.ZR), new FV(FZ.PM),new FV(FZ.PL));

    FLRS t3Table = new FLRS(new FV(FZ.FF,FZ.ZR), new FV(FZ.FF,FZ.FF), new
FV(FZ.FF,FZ.FF), new FV(FZ.FF,FZ.FF),new FV(FZ.ZR,FZ.FF));

    reader.Print();
    doubleChannelDifferentiator.Print();
    OneXOneDefaultTable.Print();
    t3Table.Print();

    DataFuzzy p0 = new DataFuzzy();
    p0.SetName("P0");
    p0.SetValue(new Fuzzy(0.0F));
    pn.PlaceList.add(p0);

    DataFuzzy p1 = new DataFuzzy(); //Room ref. temp
    p1.SetName("P1");
    pn.PlaceList.add(p1);

    DataFuzzy p2 = new DataFuzzy();
    p2.SetName("P2");
    pn.PlaceList.add(p2);

    DataFuzzy p3 = new DataFuzzy(); //from Room Plant_Current Room temp
    p3.SetName("P3");
    p3.SetValue(new Fuzzy(0.96F));
    pn.PlaceList.add(p3);

    DataFuzzy p4 = new DataFuzzy();
    p4.SetName("P4");
    pn.PlaceList.add(p4);

    DataFuzzy p5 = new DataFuzzy();
    p5.SetName("P5");
    pn.PlaceList.add(p5);

    DataFuzzy p6 = new DataFuzzy();
    p6.SetName("P6");
    pn.PlaceList.add(p6);

    DataFuzzy p7 = new DataFuzzy();
    p7.SetName("P7");
    p7.SetValue(new Fuzzy(0.0F));
    pn.PlaceList.add(p7);

    DataTransfer u = new DataTransfer();
    u.SetName("u");
    u.Value = new TransferOperation("localhost", "1081", "u"); //to Heater Tank
Plant ON/OFF
    pn.PlaceList.add(u);

```

```

DataNetworkCommand PauseCommand = new DataNetworkCommand();
PauseCommand.SetName("PauseCommand");
PauseCommand.SetValue(new NetworkCommand(NetworkCommands.Pause));
pn.ConstantPlacelist.add(PauseCommand);

DataNetworkCommand StartCommand = new DataNetworkCommand();
StartCommand.SetName("StartCommand");
StartCommand.SetValue(new NetworkCommand(NetworkCommands.Start));
pn.ConstantPlacelist.add(StartCommand);

DataFuzzy NL = new DataFuzzy();
NL.SetName("NL");
NL.SetValue(new Fuzzy(-1F));
pn.ConstantPlacelist.add(NL);

DataFuzzy PL = new DataFuzzy();
PL.SetName("PL");
PL.SetValue(new Fuzzy(1F));
pn.ConstantPlacelist.add(PL);

// T0 -----
PetriTransition t0 = new PetriTransition(pn);
t0.TransitionName = "T0";
t0.InputPlaceName.add("P0");
t0.InputPlaceName.add("P1");

Condition T0Ct1 = new Condition(t0, "P0",
TransitionCondition.NotNull);
Condition T0Ct2 = new Condition(t0, "P1",
TransitionCondition.NotNull);
T0Ct1.SetNextCondition(LogicConnector.AND, T0Ct2);

GuardMapping grdT0 = new GuardMapping();
grdT0.condition = T0Ct1;

ArrayList<PlaceNameWithWeight> input0 = new
ArrayList<>();

input0.add(new PlaceNameWithWeight("P0", 1F));
input0.add(new PlaceNameWithWeight("P1", 1F));

ArrayList<String> Output0 = new ArrayList<>();
Output0.add("P2");

grdT0.Activations.add(new Activation(t0, reader, input0,
TransitionOperation.FLRS, Output0));

t0.GuardMappingList.add(grdT0);

t0.Delay = 0;
pn.Transitions.add(t0);

```

```

// T1 -----
    PetriTransition t1 = new PetriTransition(pn);
    t1.TransitionName = "T1";
    t1.InputPlaceName.add("P2");
    t1.InputPlaceName.add("P3");

    Condition T1Ct1 = new Condition(t1, "P2",
TransitionCondition.NotNull);
    Condition T1Ct2 = new Condition(t1, "P3",
TransitionCondition.NotNull);
    T1Ct1.SetNextCondition(LogicConnector.AND, T1Ct2);

    GuardMapping grdT1 = new GuardMapping();
    grdT1.condition = T1Ct1;

    ArrayList<PlaceNameWithWeight> input1 = new ArrayList<>();
    input1.add(new PlaceNameWithWeight("P2", 1F));
    input1.add(new PlaceNameWithWeight("P3", 1F));

    ArrayList<String> Output1 = new ArrayList<>();
    Output1.add("P4");
    Output1.add("P5");

    grdT1.Activations.add(new Activation(t1,
doubleChannelDifferentiator, input1, TransitionOperation.FLRS, Output1));

    t1.GuardMappingList.add(grdT1);

    t1.Delay = 0;
    pn.Transitions.add(t1);

// T2 -----
    PetriTransition t2 = new PetriTransition(pn);
    t2.TransitionName = "T2";
    t2.InputPlaceName.add("P4");

    Condition T2Ct1 = new Condition(t2, "P4",
TransitionCondition.NotNull);

    GuardMapping grdT2 = new GuardMapping();
    grdT2.condition = T2Ct1;

    ArrayList<PlaceNameWithWeight> input2 = new ArrayList<>();
    input2.add(new PlaceNameWithWeight("P4", 1F));

    ArrayList<String> Output2 = new ArrayList<>();
    Output2.add("P0");

    grdT2.Activations.add(new Activation(t2, OneXOneDefaultTable,
input2, TransitionOperation.FLRS, Output2));

    t2.GuardMappingList.add(grdT2);

```

```

t2.Delay = 1;
pn.Transitions.add(t2);

// T3 -----
PetriTransition t3 = new PetriTransition(pn);
t3.TransitionName = "T3";
t3.InputPlaceName.add("P5");

Condition T3Ct1 = new Condition(t3, "P5",
TransitionCondition.NotNull);

GuardMapping grdT3 = new GuardMapping();
grdT3.condition = T3Ct1;

ArrayList<PlaceNameWithWeight> input3 = new ArrayList<>();
input3.add(new PlaceNameWithWeight("P5", 1F));

ArrayList<String> Output3 = new ArrayList<>();
Output3.add("P6");
Output3.add("P7");

grdT3.Activations.add(new Activation(t3, t3Table, input3,
TransitionOperation.FLRS, Output3));

t3.GuardMappingList.add(grdT3);

t3.Delay = 0;
pn.Transitions.add(t3);

// T4 -----
PetriTransition t4 = new PetriTransition(pn);
t4.TransitionName = "T4";
t4.InputPlaceName.add("P6");
t4.InputPlaceName.add("P7");

Condition T4Ct1 = new Condition(t4, "P6",
TransitionCondition.NotNull);
Condition T4Ct2 = new Condition(t4, "P6",
TransitionCondition.Equal, "NL");
T4Ct1.SetNextCondition(LogicConnector.AND, T4Ct2);

GuardMapping grdT41 = new GuardMapping();
grdT41.condition = T4Ct1;

grdT41.Activations.add(new Activation(t4, "PauseCommand",
TransitionOperation.SendOverNetwork, "u"));

t4.GuardMappingList.add(grdT41);

```

```

-----
//-----

Condition T4Ct3 = new Condition(t4, "P7",
TransitionCondition.NotNull);
Condition T4Ct4 = new Condition(t4, "P7",
TransitionCondition.Equal, "PL");
T4Ct3.SetNextCondition(LogicConnector.AND, T4Ct4);

GuardMapping grdT42 = new GuardMapping();
grdT42.condition = T4Ct3;

grdT42.Activations.add(new Activation(t4, "StartCommand",
TransitionOperation.SendOverNetwork, "u"));

t4.GuardMappingList.add(grdT42);

t4.Delay = 0;
pn.Transitions.add(t4);

// -----

pn.Delay = 500;
//pn.PrintingSpeed=0;
pn.ShowLogInWindow=true;
// pn.Start();

PetriNetWindow frame = new PetriNetWindow(false);
frame.petriNet = pn;
frame.setVisible(true);
}
}

```

#### Code Listing 5: HTC

```

package DCS_FuzzyLab.RoomHeatingSystem;

import java.io.FileNotFoundException;
import java.util.ArrayList;
import Components.Activation;
import Components.Condition;
import Components.GuardMapping;
import Components.PetriNet;
import Components.PetriNetWindow;
import Components.PetriTransition;
import DataObjects.DataFuzzy;
import DataObjects.DataTransfer;
import DataOnly.FLRS;
import DataOnly.FV;
import DataOnly.Fuzzy;
import DataOnly.PlaceNameWithWeight;
import DataOnly.TransferOperation;
import Enumerations.FZ;
import Enumerations.LogicConnector;
import Enumerations.TransitionCondition;
import Enumerations.TransitionOperation;

```

```

public class HTC {
    public static void main (String[]args) throws FileNotFoundException {
        PetriNet pn = new PetriNet();
        pn.PetriNetName = "HTC";
        pn.NetworkPort = 1083;

        pn.SetInputFile("D:\\PetriInputData\\HTC.txt");

        FLRS reader = new FLRS(new FV(FZ.NL), new FV(FZ.NM), new FV(FZ.ZR), new FV(FZ.PM), new
FV(FZ.PL),
                                                                    new FV(FZ.NL), new FV(FZ.NM), new FV(FZ.ZR), new
FV(FZ.PM), new FV(FZ.PL),
                                                                    new FV(FZ.NL), new FV(FZ.NM), new FV(FZ.ZR), new
FV(FZ.PM), new FV(FZ.PL),
                                                                    new FV(FZ.NL), new FV(FZ.NM), new FV(FZ.ZR), new
FV(FZ.PM), new FV(FZ.PL));

        FLRS doubleChannelDifferentiator = new FLRS(new FV(FZ.ZR, FZ.ZR), new FV(FZ.NM, FZ.NM), new
FV(FZ.NL, FZ.NL), new FV(FZ.NL, FZ.NL),new FV(FZ.NL, FZ.NL),
                                                                    new
FV(FZ.PM, FZ.PM), new FV(FZ.ZR, FZ.ZR), new FV(FZ.NM, FZ.NM), new FV(FZ.NL, FZ.NL),new FV(FZ.NL,
FZ.NL),
                                                                    new
FV(FZ.PL, FZ.PL), new FV(FZ.PM, FZ.PM), new FV(FZ.ZR, FZ.ZR), new FV(FZ.NM, FZ.NM),new FV(FZ.NL,
FZ.NL),
                                                                    new
FV(FZ.PL, FZ.PL), new FV(FZ.PL, FZ.PL), new FV(FZ.PM, FZ.PM), new FV(FZ.ZR, FZ.ZR),new FV(FZ.NM,
FZ.NM),
                                                                    new
FV(FZ.PL, FZ.PL), new FV(FZ.PL, FZ.PL), new FV(FZ.PL, FZ.PL), new FV(FZ.PM, FZ.PM),new FV(FZ.ZR,
FZ.ZR));

        FLRS doubleChannelAdder = new FLRS(new FV(FZ.NL, FZ.NL), new FV(FZ.NL, FZ.NL), new FV(FZ.NL,
FZ.NL), new FV(FZ.NM, FZ.NM),new FV(FZ.ZR, FZ.ZR),
                                                                    new FV(FZ.NL, FZ.NL), new
FV(FZ.NL, FZ.NL), new FV(FZ.NM, FZ.NM), new FV(FZ.ZR, FZ.ZR),new FV(FZ.PM, FZ.PM),
                                                                    new FV(FZ.NL, FZ.NL), new
FV(FZ.NM, FZ.NM), new FV(FZ.ZR, FZ.ZR), new FV(FZ.PM, FZ.PM),new FV(FZ.PL, FZ.PL),
                                                                    new FV(FZ.NM, FZ.NM), new
FV(FZ.ZR, FZ.ZR), new FV(FZ.PM, FZ.PM), new FV(FZ.PL, FZ.PL),new FV(FZ.PL, FZ.PL),
                                                                    new FV(FZ.ZR, FZ.ZR), new
FV(FZ.PM, FZ.PM), new FV(FZ.PL, FZ.PL), new FV(FZ.PL, FZ.PL),new FV(FZ.PL, FZ.PL));

        FLRS OneXOneDefaultTable = new FLRS(new FV(FZ.NL), new FV(FZ.NM), new FV(FZ.ZR), new
FV(FZ.PM),new FV(FZ.PL));

        reader.Print();
        doubleChannelDifferentiator.Print();
        doubleChannelAdder.Print();
        OneXOneDefaultTable.Print();

        DataFuzzy p0 = new DataFuzzy();
        p0.SetName("P0");
        p0.SetValue(new Fuzzy(0.0F));
        pn.PlaceList.add(p0);

        DataFuzzy p1 = new DataFuzzy(); //Reference WaterTemp
        p1.SetName("P1");
        pn.PlaceList.add(p1);

        DataFuzzy p2 = new DataFuzzy();
        p2.SetName("P2");
        pn.PlaceList.add(p2);
    }
}

```

```

DataFuzzy p3 = new DataFuzzy(); //from HeaterTank_Current WaterTemp
p3.SetName("P3");
p3.SetValue(new Fuzzy(0.3F));
pn.PlaceList.add(p3);

DataFuzzy p4 = new DataFuzzy();
p4.SetName("P4");
pn.PlaceList.add(p4);

DataFuzzy p5 = new DataFuzzy();
p5.SetName("P5");
pn.PlaceList.add(p5);

DataFuzzy p6 = new DataFuzzy();
p6.SetName("P6");
p6.SetValue(new Fuzzy(0.0F));
pn.PlaceList.add(p6);

DataFuzzy p7 = new DataFuzzy();
p7.SetName("P7");
pn.PlaceList.add(p7);

DataFuzzy p8 = new DataFuzzy();
p8.SetName("P8");
pn.PlaceList.add(p8);

DataTransfer gascmd = new DataTransfer();
gascmd.SetName("gascmd");
gascmd.Value = new TransferOperation("localhost", "1081", "u"); //to HeaterTank Plant
pn.PlaceList.add(gascmd);

// T0 -----
PetriTransition t0 = new PetriTransition(pn);
t0.TransitionName = "T0";
t0.InputPlaceName.add("P0");
t0.InputPlaceName.add("P1");

Condition T0Ct1 = new Condition(t0, "P0", TransitionCondition.NotNull);
Condition T0Ct2 = new Condition(t0, "P1", TransitionCondition.NotNull);
T0Ct1.SetNextCondition(LogicConnector.AND, T0Ct2);

GuardMapping grdT0 = new GuardMapping();
grdT0.condition = T0Ct1;

ArrayList<PlaceNameWithWeight> input0 = new ArrayList<>();
input0.add(new PlaceNameWithWeight("P0", 1F));
input0.add(new PlaceNameWithWeight("P1", 1F));

ArrayList<String> Output0 = new ArrayList<>();
Output0.add("P2");

grdT0.Activations.add(new Activation(t0, reader, input0,
TransitionOperation.FLRS, Output0));

t0.GuardMappingList.add(grdT0);

t0.Delay = 0;
pn.Transitions.add(t0);

// T1 -----
PetriTransition t1 = new PetriTransition(pn);
t1.TransitionName = "T1";
t1.InputPlaceName.add("P2");

```

```

t1.InputPlaceName.add("P3");

Condition T1Ct1 = new Condition(t1, "P2", TransitionCondition.NotNull);
Condition T1Ct2 = new Condition(t1, "P3", TransitionCondition.NotNull);
T1Ct1.SetNextCondition(LogicConnector.AND, T1Ct2);

GuardMapping grdT1 = new GuardMapping();
grdT1.condition = T1Ct1;

ArrayList<PlaceNameWithWeight> input1 = new ArrayList<>();
input1.add(new PlaceNameWithWeight("P2", 1F));
input1.add(new PlaceNameWithWeight("P3", 1F));

ArrayList<String> Output1 = new ArrayList<>();
Output1.add("P4");
Output1.add("P5");

grdT1.Activations.add(new Activation(t1, doubleChannelDifferentiator, input1,
TransitionOperation.FLRS, Output1));

t1.GuardMappingList.add(grdT1);

t1.Delay = 0;
pn.Transitions.add(t1);

// T2 -----
PetriTransition t2 = new PetriTransition(pn);
t2.TransitionName = "T2";
t2.InputPlaceName.add("P7");

Condition T2Ct1 = new Condition(t2, "P7", TransitionCondition.NotNull);

GuardMapping grdT2 = new GuardMapping();
grdT2.condition = T2Ct1;

grdT2.Activations.add(new Activation(t2, "P7",
TransitionOperation.SendOverNetwork, "gascmd"));

t2.GuardMappingList.add(grdT2);

t2.Delay = 0;
pn.Transitions.add(t2);

// T3 -----
PetriTransition t3 = new PetriTransition(pn);
t3.TransitionName = "T3";
t3.InputPlaceName.add("P5");

Condition T3Ct1 = new Condition(t3, "P5", TransitionCondition.NotNull);

GuardMapping grdT3 = new GuardMapping();
grdT3.condition = T3Ct1;

ArrayList<PlaceNameWithWeight> input3 = new ArrayList<>();
input3.add(new PlaceNameWithWeight("P5", 1F));

ArrayList<String> Output3 = new ArrayList<>();
Output3.add("P0");

grdT3.Activations.add(new Activation(t3, OneXOneDefaultTable, input3,
TransitionOperation.FLRS, Output3));

t3.GuardMappingList.add(grdT3);

```

```

t3.Delay = 1;
pn.Transitions.add(t3);

// T4 -----
PetriTransition t4 = new PetriTransition(pn);
t4.TransitionName = "T4";
t4.InputPlaceName.add("P4");
t4.InputPlaceName.add("P6");

Condition T4Ct1 = new Condition(t4, "P4", TransitionCondition.NotNull);
Condition T4Ct2 = new Condition(t4, "P6", TransitionCondition.NotNull);
T4Ct1.SetNextCondition(LogicConnector.AND, T4Ct2);

GuardMapping grdT4 = new GuardMapping();
grdT4.condition = T4Ct1;

ArrayList<PlaceNameWithWeight> input4 = new ArrayList<>();
input4.add(new PlaceNameWithWeight("P4", 1.2F));
input4.add(new PlaceNameWithWeight("P6", 1F));

ArrayList<String> Output4 = new ArrayList<>();
Output4.add("P7");
Output4.add("P8");

grdT4.Activations.add(new Activation(t4, doubleChannelAdder, input4,
TransitionOperation.FLRS, Output4));

t4.GuardMappingList.add(grdT4);

t4.Delay = 1;
pn.Transitions.add(t4);

// T5 -----
PetriTransition t5 = new PetriTransition(pn);
t5.TransitionName = "T5";
t5.InputPlaceName.add("P8");
t5.InputPlaceName.add("P6");

Condition T5Ct1 = new Condition(t5, "P8", TransitionCondition.NotNull);

GuardMapping grdT5 = new GuardMapping();
grdT5.condition = T5Ct1;

ArrayList<PlaceNameWithWeight> input5 = new ArrayList<>();
input5.add(new PlaceNameWithWeight("P8", 1F));

ArrayList<String> Output5 = new ArrayList<>();
Output5.add("P6");

grdT5.Activations.add(new Activation(t5, OneXOneDefaultTable, input5,
TransitionOperation.FLRS, Output5));

t5.GuardMappingList.add(grdT5);

t5.Delay = 0;
pn.Transitions.add(t5);

// -----

System.out.println("HTC started \n -----");
pn.Delay = 500;

```

```

        //pn.PrintingSpeed=0;
        pn.ShowLogInWindow=true;
        // pn.Start();

        PetriNetWindow frame = new PetriNetWindow(false);
        frame.petriNet = pn;
        frame.setVisible(true);
    }
}

```

## Exercises:

1. Test the application for different scenarios.
2. The HTC component controller is constructed on the P model. Starting from this, build PI model of the HTC
3. Starting from the above application, build an application that contains another component called the Outside Reference Calculator (ORC), which has an input port to read the outside temperature and based on that tries to estimate the optimum water temperature in the boiler. The diagram of the components in Figure 4.1 is presented, and in Figure 4.2 is presented the Petri net that model the component.

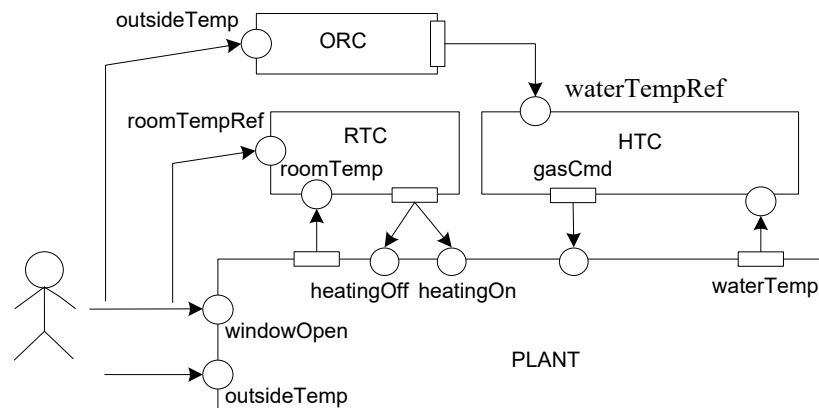


Figure 4.1 Component diagram

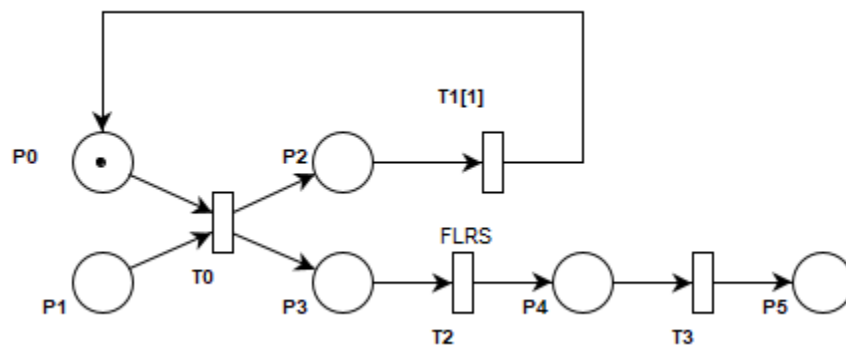


Figure 4.2 Petri net of the ORC component

## 5. Knowledge veirfication

1. What are the system components and what role do they have?
2. How does the performance of the system is influenced by the weights of the arcs?
3. What are the instructions that implement the component ports?
4. How to attach FLRS tables to transitions using the All\_Petri\_Nets\_Framework?