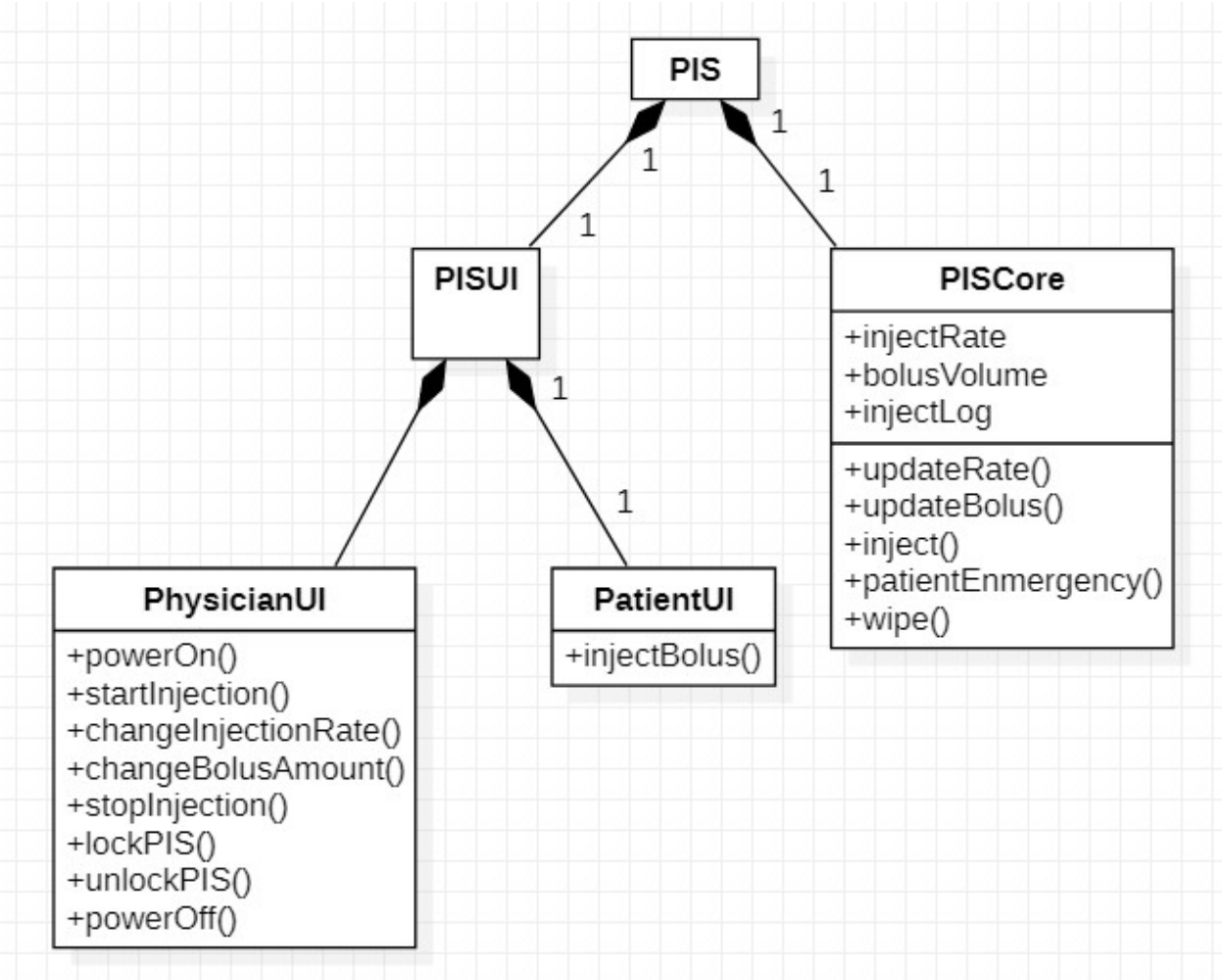
SOFTWARE VALIDATION

Painkiller Injection System

Table of Contents

System Architecture

The system architecture is shown below:



T1: Unit Test

T1.1: Injector Unit Test

T1.1.1: Test injector(a)

```
function obj =injector(a) Tcover1.1.1
      obj.inject_rate = 0.01;
      obj.bolus_volume = 0.2;
      obj.inject_log = [];
end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.1.1
Coverage Item	Tcover1.1.1
Input	Null
State	Initialization of the injector system
Expected Output	obj =injector(a)

- Test coverage: 1/1=100%
- Test result: 1 passed

T1.1.2: Test update_rate ()

```
function update_rate(this,new_rate) Tcover1.1.2
      this.inject_rate = new_rate;
end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.1.2
Coverage Item	Tcover1.1.2
Input	• Injector system • A target rate
State	An update of baseline injecting rate
Expected Output	• Initialized injector • The injector with baseline rate updated

- Test coverage: 1/1=100%
- Test result: 1 passed

T1.1.3: Test update_bolus ()

```
function update_bolus(this,new_bolus) Tcover1.1.3
      this.bolus_volume = new_bolus;
end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.1.3
Coverage Item	Tcover1.1.3

Input	• Injector system • A target bolus volume
State	an update of bolus injecting rate
Expected Output	• Initialized injector • The injector with bolus volume updated

- Test coverage: 1/1=100%
- Test result: 1 passed

T1.1.4: Test injection ()

```
function inject(this,little) Tcover1.1.4
      this.inject_log(end+1) = little;
end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.1.4
Coverage Item	Tcover1.1.4
Input	• Injector system
State	An injection happens at current moment
Expected Output	• Initialized injector • A log that has recorded the most recent injection

- Test coverage: 1/1=100%
- Test result: 1 passed

T1.1.5: Test bolus ()

```
function patient_enmergency(this,vol) Tcover1.1.5
      this.inject_log(end+1) = vol;
end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.1.5
Coverage Item	Tcover1.1.5
Input	• Injector system • A bolus input
State	A bolus shot happens at current moment
Expected Output	• Initialized injector • A log that has recorded the most recent bolus

- Test coverage: 1/1=100%
- Test result: 1 passed

T1.1.6: Test timefly ()

```
function timefly(this) Tcover1.1.6
      this.inject_log(end+1) = 0;
```

```
end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.1.6
Coverage Item	Tcover1.1.6
Input	• Injector system
State	No action in the current moment
Expected Output	• Initialized injector • A log that has a 0 at the end

- Test coverage: 1/1=100%
- Test result: 1 passed

T1.1.7: Test wipe ()

```
function wipe(this) Tcover1.1.7
      this.inject_rate = 0.01;
      this.bolus_volume = 0.2;
      this.inject_log =[];
end
```

Coverage Criteria: Statement coverage

- Test case

	Test Case T1.1.7
Coverage Item	Tcover1.1.7
Input	• Injector system • A current baseline rate and current bolus volume
State	Need to reset the whole system
Expected Output	• Initialized injector • Cleared log • Default injection rate • Default bolus volume

- Test coverage: 1/1=100%
- Test result: 1 passed

T1.2: PISdisplay Unit Test

This Unit Test tests some of the logic functions written in PISdisplay. However, some logical calculation related operation are written in the callback functions of the items, so they will be not tested here but rather in the UItest

For the sake of safety and security, we have set some core functions and properties to private in PISdisplay. However, as we are doing a white box test and for efficiency tests, please modify the following properties and methods to "public" in order for the testcase to access.

```
properties (Access = private)
      sf % stateflow function
      cc % instance for class
      Patient_UI % multiwindow
```

```
end

methods (Access = private)
        ……
        function live(app,~,~)
                ……
        end
        ……
end
```

T1.2.1: Test live()

```
function live(app,~,~)
        if length(app.cc.inject_log) > 3600 &&(app.HourAmount.Value >=0)&&(strcmp(app.PowerSwitch.Value,'On')) Tcover1.2.1.1
                app.HourAmountGauge.Value = app.HourAmountGauge.Value - app.cc.inject_log(length(app.cc.inject_log)-3600);
                app.HourAmount.Value = app.HourAmountGauge.Value;
        end
        if length(app.cc.inject_log) > 86400 && (app.DayAmount.Value >= 0)&& (strcmp(app.PowerSwitch.Value,'On')) Tcover1.2.1.2
                app.DayamountGauge.Value = app.DayamountGauge.Value - app.cc.inject_log(length(app.cc.inject_log)- 86400);
                app.DayAmount.Value = app.DayamountGauge.Value;
        end
        if strcmp(app.InjectionSwitch.Value,'On') && (app.DayAmount.Value < 3 ) && (app.HourAmount.Value < 1)&&
        (strcmp(app.PowerSwitch.Value,'On')) Tcover1.2.1.3
                app.cc.inject(app.cc.inject_rate/60)
                app.DayamountGauge.Value = app.DayamountGauge.Value + app.cc.inject_log(end);
                app.DayAmount.Value = app.DayamountGauge.Value;
                app.HourAmountGauge.Value = app.HourAmountGauge.Value + app.cc.inject_log(end);
                app.HourAmount.Value = app.HourAmountGauge.Value;
        else Tcover1.2.1.4
                app.cc.timefly
        end
        if ((app.DayAmount.Value >= 3 ) || (app.HourAmount.Value >= 1))&& (strcmp(app.PowerSwitch.Value,'On')) Tcover1.2.1.5
                app.powerlight.Color = [1,1,0];
        else Tcover1.2.1.6
                app.powerlight.Color = [0,1,0];
        end
        app.TimesEditField.Value = app.TimesEditField.Value + 1;

 end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.2.1.1	Test Case T1.2.1.2	Test Case T1.2.1.3
Coverage Item	Tcover1.2.1.1 Tcover1.2.1.2 Tcover1.2.1.5	Tcover1.2.1.4 Tcover1.2.1.6	Tcover1.2.1.3
Input	• PISdisplay • an inject_log length>86400 with all 0.02 • houramount = dayamount= 3600*0.02 • power is on • injection switch is off	• PISdisplay • an inject_log length>=10 with all 0.02 • houramount = dayamount= 0 • power is on • injection switch is off	• PISdisplay • an inject_log length>=10 with all 0.02 • houramount = dayamount= 0.2 • power is on • injection switch is on • injection rate=0.02*60
State	injecting over 1 day and overamount	Normal condition and no current injection	Normal injection and normal injection
Expected Output	• houramountgauge=3599*0.02 • dayamountgauge=3599*0.02 • the indicating light turns yellow	• log(end)==0.02 • the indicating light turns green	• log(end)==0.02 • HourAmountGauge==0.22 • DayamountGauge==0.22

- Test coverage: 3/3=100%
- Test result: 3 passed

Test1.2.2: test wipe()

```
function wipe(this) Tcover1.2.2
      try stop(timerfindall);    catch; end
      try delete(timerfindall);  catch; end
      app.HourAmount.Value = 0;
      app.DayAmount.Value = 0;
      app.DayamountGauge.Value = 0;
      app.HourAmountGauge.Value = 0;
      app.TimesEditField.Value = 0;
      app.baselinerate.Value = 0.01;
      app.PerBolusKnob.Value = 0.2;
      app.InjectionSwitch.Value = 'Off';
      app.cc.wipe;
      try app.Patient_UI.close; catch; end
      app.TimeScalingSlider.Value = 1;
end
```

Coverage Criteria: Statement coverage

- Test case

	Test Case T1.2.2
Coverage Item	Tcover1.2.2
Input	• PISdisplay • an inject_log length>=10 with all 0.02 • houramount = dayamount= 0.2 • power is on • injection switch is on • injection rate=0.02*60
State	Need to reset the system when the power is switched off and call wipe()
Expected Output	• Initialized injector • log(end)==[] • HourAmountGauge==0 • DayamountGauge==0

- Test coverage: 1/1=100%
- Test result: 1 passed

Test1.2.3: test wipe()

```
function patient(this) Tcover1.2.3
      if app.HourAmount.Value + app.PerBolusKnob.Value < 1 &&
app.DayAmount.Value + app.PerBolusKnob.Value < 3
            app.cc.patient_enmergency(app.cc.bolus_volume)
            app.DayamountGauge.Value = app.DayamountGauge.Value +
      app.cc.bolus_volume;
            app.DayAmount.Value = app.DayamountGauge.Value;
            app.HourAmountGauge.Value = app.HourAmountGauge.Value +
      app.cc.bolus_volume;
            app.HourAmount.Value = app.HourAmountGauge.Value;
      end
end
```

Coverage Criteria: Statement coverage

- Test case

	Test Case T1.2.3
Coverage Item	Tcover1.2.3
Input	• PISdisplay • an inject_log length>=10 with all 0.02 • houramount = dayamount= 0.2 • power is on
State	To inject a bolus for patients
Expected Output	• log==[end] == bolus_volume • dayamount = 0.02+bolus_volume • houramount=0.02+bolus_volume

- Test coverage: 1/1=100%
- Test result: 1 passed

T2: Integration Test

The implementation only contains UI and one core class name “Injector”, thus the validation will skip this part and directly continue with Functional Test

T3: Functional Test

T3.1: Use Case "Power On"

The Physician turns on the injector.

T3.2: Use Case "Power Off"

The Physician turns off the injector.

- Test case

	Test Case TC3.1
Coverage Item	T3.1 T3.2
Input	• PISdisplay
State	To turn on and turn off the machine
Expected Output	• The machine turns on first • The machine turns off

- Test result: 1 passed

T3.3: Use Case "Start Injection"

T3.4: Use Case "Stop Injection"

The Physician turns on the injector to start injecting. He also changes the baseline rate in the middle. The Physician turns off the injector after the injecting. During the process, the there maybe over injecting from either baseline injection or from bolus injection.

- Test case

	Test Case TC3.4 TC3.7 TC3.8
Coverage Item	T3.3 T3.4
Input	• PISdisplay • Turn on the injection switch
State	To turn on and turn off the machine
Expected Output	• The machine turns on first. • The machine starts to inject. • The machine stops injecting.

- Test result: 1 passed

T3.5: Use Case "Change baseline rate"

The Physician tries to change the baseline injection rate and tends to change it to over 0.1

- Test case

	Test Case TC3.2
Coverage Item	T3.5
Input	• PISdisplay • Changes to baseline rate
State	Need to change baseline rate
Expected Output	• The machine turns on first. • The baseline rate can be changed for the first time

	• The baseline cannot be changed for the second time

- Test result: 1 passed

T3.6: Use Case "Change bolus amount"

The Physician tries to change the bolus volume and tends to change it to over 0.5

- Test case

	Test Case TC3.3
Coverage Item	T3.6
Input	• PISdisplay • Changes to bolus volume
State	Need to change bolus volume
Expected Output	• The machine turns on first. • The bolus volume can be changed for the first time • The bolus volume cannot be changed for the second time

- Test result: 1 passed

T3.7: Use Case "Lock"

T3.8: Use Case "Unlock"

The Physician locks the machine so that no buttons on the machines can be changed except for the patient button to shot bolus.

- Test case

	Test Case TC3.5 TC3.6
Coverage Item	T3.7 T3.8
Input	• PISdisplay • Locks the machine • Patient injects bolus
State	Need to lock the machine
Expected Output	• The machine turns on first. • The machine is locked so that nothing can be modified • The patient button is still workable.

- Test result: 1 passed

T3.9: Use Case "Inject Bolus"

Patients want to inject a bolus of painkiller for himself to kill the pain no matter if the machine is locked.

- Test case

	Test Case TC3.6
Coverage Item	T3.9
Input	• PISdisplay • Locks the machine

	• Patient injects bolus
State	Need to lock the machine
Expected Output	• The machine turns on first. • The machine is locked so that nothing can be modified • The patient button is still workable.

- Test result: 1 passed

T4: UPPAAL Model Checking

This UPPAAL runs under Windows10 with UPPAAL 4.1.24.

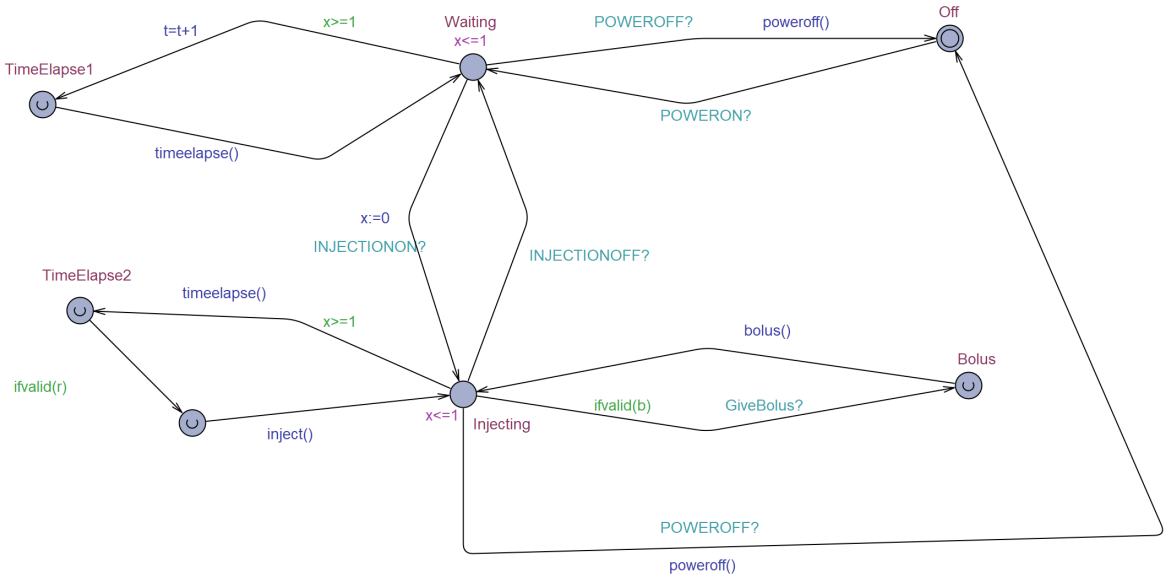
Since UPPAAL weakly supports Float data type, all the real data required has been multiplied by 10.

Further, since the baseline input rate and the bolus volume does not impact the logic and the structure of the system, the model used r=1 and b=20 to resemble the input, respectively.

T4.1 PIS Model

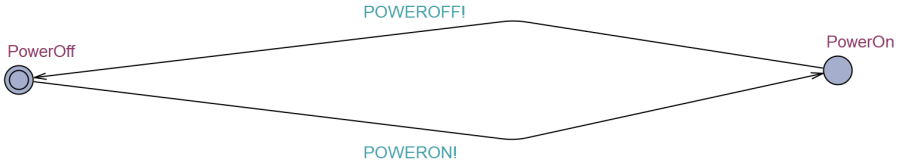
This resembles the implementation of Matlab painkiller system. Most of the core functions are re-implemented in UPPAAL again.

The core function is *ifvalid(input_amount)*, which checks if a new input of input_amount will violate the limit of input both daily and hourly.



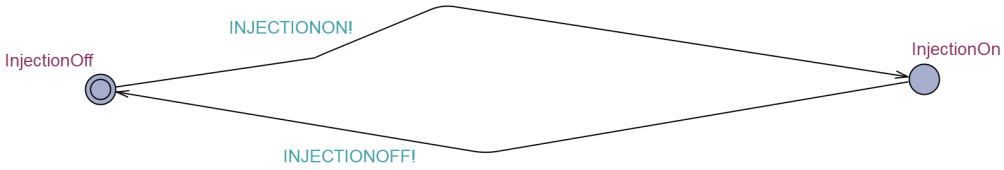
T4.2 PowerButton Model

The POWERON and POWEROFF resembles the push action of the power switch.



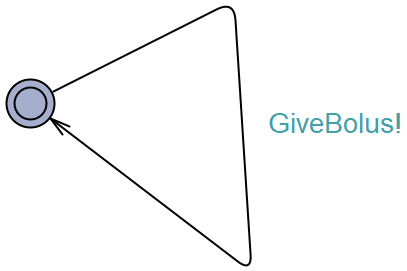
T4.3 InjectionButton Model

The INJECTIONON and INJECTIONOFF resembles the push action of real injection switch

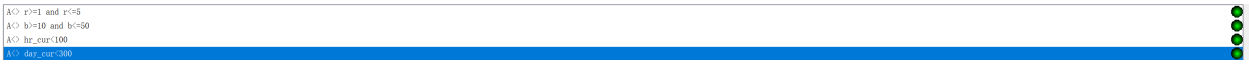


T4.4 PatientButton Model

The GiveBolus resembles the push of patient button by the patients to get an extra bolus input.



T4.5 Properties Checking



These properties checking mainly focus on the limits of injection of baseline and bolus to see if there will be a mis-overshoot. It turns out that the system is safe and sound.