Group 8

Author: Mingzhao Guo

Software Validation

Painkiller Injector

Table of Contents

[System Architecture 2](#_Toc44237249)

[T1: Unit Test 2](#_Toc44237250)

[T1.1: InjectorDB Unit Test 2](#_Toc44237251)

[T1.2: InjectorProcessor Unit Test 4](#_Toc44237252)

[T2: Integration Test 12](#_Toc44237253)

[T2.1: OrderProcessor+OrderDB Integration 12](#_Toc44237254)

[T2.2: ServerUI+OrderProcessor+OrderDB Integration 14](#_Toc44237255)

[T3: Functional Test 14](#_Toc44237256)

[T4: Modal checking 14](#_Toc44237257)

## System Architecture

The system architecture is shown below:

电脑屏幕截图

描述已自动生成

## T1: Unit Test

### T1.1: InjectorDB Unit Test

T1.1.1: Test SetAmountLimit()

function SetAmountLimit(DB,data)

DB.AmountLimit=data;

end

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.1.1.1 |
| Coverage Item | Tcover1.1.1.1 |
| Input | data = 1; |
| State | db=InjectorDB; |
| Expected Output | db.SetAmountLimit(data); |

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

T1.1.2: Test SetAmountInShortPeriod()

function SetAmountInShortPeriod(DB,data)

DB.AmountInShortPeriod=data;

end

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.1.2.1 |
| Coverage Item | Tcover1.1.2.1 |
| Input | data = 1; |
| State | db=InjectorDB; |
| Expected Output | db.SetAmountInShortPeriod(data); |

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

T1.1.3: Test SetBaseline ()

function SetBaseline(DB,data)

DB.Baseline=data;

end

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.1.3.1 |
| Coverage Item | Tcover1.1.3.1 |
| Input | data = 1; |
| State | db=InjectorDB; |
| Expected Output | db.SetBaseline(data); |

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

T1.1.4: Test SetBolus ()

function SetBolus(DB,data)

DB.Bolus=data;

end

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.1.4.1 |
| Coverage Item | Tcover1.1.4.1 |
| Input | data = 1; |
| State | db=InjectorDB; |
| Expected Output | db.SetBolus(data); |

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

T1.1.5: Test SetAuthority ()

function SetAuthority(DB,bol)

DB.Authority=bol;

end

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.1.5.1 |
| Coverage Item | Tcover1.1.5.1 |
| Input | data = 1; |
| State | db=InjectorDB; |
| Expected Output | db.SetAuthority(data); |

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

T1.1.6: Test UpdateTotalAmount ()

function UpdateTotalAmount(DB,data)

DB.TotalAmount=data;

end

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.1.6.1 |
| Coverage Item | Tcover1.1.6.1 |
| Input | data = 1; |
| State | db=InjectorDB; |
| Expected Output | db.UpdateTotalAmount(data); |

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

T1.2.1: Test checkSpeed ()

function re = checkSpeed(~,data) %check baseline

value = str2double(data);

if ((value <= 0.1) && (value >=0.01))

re = 't';

else

re = 'f';

end

end

* Coverage Criteria: Branch coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.2.1.1 |
| Coverage Item | Tcover1.2.1.1 |
| Input | data1 = '0.05'; |
| State | pro=InjectorProcessor; |
| Expected Output | pro.checkSpeed(data1)=’t’; |

|  |  |
| --- | --- |
|  | Test Case T1.2.1.2 |
| Coverage Item | Tcover1.2.1.2 |
| Input | data2 = '0.2'; |
| State | pro=InjectorProcessor; |
| Expected Output | pro.checkSpeed(data2)=’f’; |

|  |  |
| --- | --- |
|  | Test Case T1.2.1.3 |
| Coverage Item | Tcover1.2.1.3 |
| Input | Data3 = '0.001'; |
| State | pro=InjectorProcessor; |
| Expected Output | pro.checkSpeed(data3)=’f’; |

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

T1.2.2: Test checkBolus ()

function re = checkBolus(~,data) %check bolus

value = str2double(data);

if (value >= 0.2 && value <=0.5)

re = 't';

else

re = 'f';

end

end

* Coverage Criteria: Branch coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.2.2.1 |
| Coverage Item | Tcover1.2.2.1 |
| Input | data1 = '0.3'; |
| State | pro=InjectorProcessor; |
| Expected Output | pro.checkBolus(data1)=’t’; |

|  |  |
| --- | --- |
|  | Test Case T1.2.2.2 |
| Coverage Item | Tcover1.2.2.2 |
| Input | data2 = '0.1'; |
| State | pro=InjectorProcessor; |
| Expected Output | pro.checkBolus(data2)=’f’; |

|  |  |
| --- | --- |
|  | Test Case T1.2.2.3 |
| Coverage Item | Tcover1.2.2.3 |
| Input | data3 = '0.6'; |
| State | pro=InjectorProcessor; |
| Expected Output | pro.checkBolus(data3)=’f’; |

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

T1.2.3: Test updateTotalAmount ()

function updateTotalAmount(process,temp)

process.InjectorDB.TotalAmount = temp;

tempstr = num2str(temp,'%.2f');

tempstr = strcat(tempstr,' ml ');

tempstr = strcat('TotalAmount:',tempstr);

process.App.DisplayTextArea.Value = tempstr;

end

* Coverage Criteria: statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.2.3.1 |
| Coverage Item | Tcover1.2.3.1 |
| Input | data1 = '1'; |
| State | db = InjectorDB; |
| Expected Output | db.TotalAmount=data1 |

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

T1.2.4: Test CaculateHour ()

function CaculateHour(process,~,~)

if strcmp(get(process.t, 'Running'),'on')

process.HourCache(process.i) = str2double(process.InjectorDB.Baseline)/60;

else

process.HourCache(process.i) = 0;

end

sum = 0.0;

for a=1:3600

sum=process.HourCache(a)+sum;

end

if ((sum>=str2double(process.InjectorDB.AmountInShortPeriod))&&(process.timerstateHour ~= 1))

process.timerstateHour = 1;

process.interupt=1;

stop(process.t);

stop(process.m);

if strcmp(get(process.e, 'Running'),'on')

stop(process.e);

process.p = 0;

end

end

if ((sum < str2double(process.InjectorDB.AmountInShortPeriod)) && (process.timerstateHour == 1))

start(process.t);

start(process.m);

if (process.p == 0)

start(process.e);

process.p = 1;

end

process.timerstateHour = 2;

end

if(process.i <3600)

process.i=process.i+1;

else

process.i = 1;

end

end

* Coverage Criteria: Branch coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.2.4.1 |
| Coverage Item | Tcover1.2.4.1 |
| Input | db.Baseline = '0.1';  db.Bolus = '0.5'; pro.HourCache(3011:3600)=0.0017; |
| State | start(pro.t);  start(pro.caculateHour); |
| Expected Output | pro.interupt=1 |

|  |  |
| --- | --- |
|  | Test Case T1.2.4.2 |
| Coverage Item | Tcover1.2.4.2 |
| Input | db.Baseline = '0.1';  db.Bolus = '0.5'; |
| State | start(pro.t);  start(pro.caculateHour); |
| Expected Output | pro.interupt=0 |

|  |  |
| --- | --- |
|  | Test Case T1.2.4.3 |
| Coverage Item | Tcover1.2.4.3 |
| Input | db.Baseline = '0.1';  db.Bolus = '0.5'; |
| State | start(pro.caculateHour); |
| Expected Output | pro.interupt=0 |

|  |  |
| --- | --- |
|  | Test Case T1.2.4.4 |
| Coverage Item | Tcover1.2.4.4 |
| Input | db.Baseline = '0.1';  db.Bolus = '0.5'; |
| State | start(pro.t);  start(pro.e);  start(pro.caculateHour); |
| Expected Output | pro.interupt=0 |

* Test coverage:4/4=100%
* Test result: 4 passed

T1.2.5: Test CaculateDay ()

function CaculateDay(process,~,~)

if strcmp(get(process.t, 'Running'),'on')

process.DayCache(process.j) = str2double(process.InjectorDB.Baseline)/60;

else

process.DayCache(process.j) = 0;

end

sum = 0;

for a=1:86400

sum=process.DayCache(a)+sum;

end

if ((sum >= str2double(process.InjectorDB.AmountLimit))&&(process.timerstateDay ~= 1))

stop(process.t);

stop(process.m);

process.interupt=1;

if strcmp(get(process.e, 'Running'),'on')

stop(process.e);

process.q = 0;

end

process.timerstateDay = 1;

end

if ((sum < str2double(process.InjectorDB.AmountLimit)) && (process.timerstateDay == 1))

start(process.t);

start(process.m);

if (process.q == 0)

start(process.e);

process.q = 1;

end

process.timerstateDay = 2;

end

if(process.j <86400)

process.j=process.j+1;

else

process.j = 1;

end

end

* Coverage Criteria: Branch coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.2.5.1 |
| Coverage Item | Tcover1.2.5.1 |
| Input | db.Baseline = '0.1';  db.Bolus = '0.5'; pro.DayCache(84635:86400)=0.0017; |
| State | start(pro.t);  start(pro.caculateDay); |
| Expected Output | pro.interupt=1 |

|  |  |
| --- | --- |
|  | Test Case T1.2.5.2 |
| Coverage Item | Tcover1.2.5.2 |
| Input | db.Baseline = '0.1';  db.Bolus = '0.5'; |
| State | start(pro.t);  start(pro.caculateDay); |
| Expected Output | pro.interupt=0 |

|  |  |
| --- | --- |
|  | Test Case T1.2.5.3 |
| Coverage Item | Tcover1.2.5.3 |
| Input | db.Baseline = '0.1';  db.Bolus = '0.5'; |
| State | start(pro.caculateDay); |
| Expected Output | pro.interupt=0 |

|  |  |
| --- | --- |
|  | Test Case T1.2.5.4 |
| Coverage Item | Tcover1.2.5.4 |
| Input | db.Baseline = '0.1';  db.Bolus = '0.5'; |
| State | start(pro.t);  start(pro.e);  start(pro.caculateDay); |
| Expected Output | pro.interupt=0 |

* Test coverage:4/4=100%
* Test result: 4 passed

T1.2.6: Test CaculateBolusforEm ()

function CaculateBolusforEm(process,~,~)

if (process.temp2 < str2double(process.InjectorDB.BolusforEm))

process.temp2 = process.temp2 + str2double(process.InjectorDB.BaselineforEm)/60;

else

process.InjectorDB.Baseline = num2str(str2double(process.InjectorDB.Baseline)-str2double(process.InjectorDB.BaselineforEm))

process.temp2 = 0;

process.App2.emergencyshotButton.Enable = 'on';

if strcmp(get(process.m, 'Running'),'off')

process.App.setButton.Enable = 'on';

process.App.startButton.Enable = 'on';

process.App.TextArea\_BaselineforEm.Value = 'waiting!';

process.App.TextArea\_BolusforEm.Value = 'waiting!';

process.App.TextArea\_Baseline.Value = 'waiting!';

process.App.TextArea\_Bolus.Value = 'waiting!';

process.InjectorDB.Bolus = 'empty';

process.InjectorDB.BaselineforEm = 'empty';

process.InjectorDB.BolusforEm = 'empty';

process.InjectorDB.TotalAmount = 0.0;

stop(process.t);

end

stop(process.e);

end

end

* Coverage Criteria: Branch coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.2.6.1 |
| Coverage Item | Tcover1.2.6.1 |
| Input | db.Baseline = '0.1';  db.Bolus = '0.5';  db.BolusforEm = '0.1';  db.BaselineforEm = '0.06'; |
| State | start(pro.t); |
| Expected Output | pro.temp2=0.001 |

|  |  |
| --- | --- |
|  | Test Case T1.2.6.2 |
| Coverage Item | Tcover1.2.6.2 |
| Input | db.Baseline = '0.1';  db.Bolus = '0.5';  db.BolusforEm = '0'; db.BaselineforEm = '0.06'; |
| State | start(pro.t); |
| Expected Output | pro.temp=2,0 |

* Test coverage:2/2=100%
* Test result: 2 passed

T1.2.7: Test CaculateBolus ()

function CaculateBolus(process,~,~)

if (process.temp3 < process.InjectorDB.BolusOrigin)

process.temp3 = process.temp3 + process.InjectorDB.BaselineOrigin/60;

else

process.InjectorDB.Baseline = num2str(str2double(process.InjectorDB.Baseline)-process.InjectorDB.BaselineOrigin);

process.temp3 = 0;

if strcmp(get(process.e, 'Running'),'off')

process.App.setButton.Enable = 'on';

process.App.startButton.Enable = 'on';

process.App.TextArea\_BaselineforEm.Value = 'waiting!';

process.App.TextArea\_BolusforEm.Value = 'waiting!';

process.App.TextArea\_Baseline.Value = 'waiting!';

process.App.TextArea\_Bolus.Value = 'waiting!';

process.InjectorDB.Bolus = 'empty';

process.InjectorDB.BaselineforEm = 'empty';

process.InjectorDB.BolusforEm = 'empty';

process.InjectorDB.TotalAmount = 0.0;

stop(process.t);

end

stop(process.m);

end

end

* Coverage Criteria: Branch coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.2.7.1 |
| Coverage Item | Tcover1.2.7.1 |
| Input | db.BolusOrigin = 0.1;  db.BaselineOrigin = 0.06; |
| State | db = InjectorDB;  start(pro.t); |
| Expected Output | pro.temp3=0.001 |

|  |  |
| --- | --- |
|  | Test Case T1.2.7.2 |
| Coverage Item | Tcover1.2.7.2 |
| Input | pro.temp3 = 1;  db.Baseline = '0.1';  db.BolusOrigin = 0.05; db.BaselineOrigin = 0.05; |
| State | db = InjectorDB;  start(pro.t); |
| Expected Output | pro.temp3=0  db.Baseline='0.05' |

* Test coverage:2/2=100%
* Test result: 2 passed

## T2: Integration Test

### T2.1: InjectorProcessor+InjectorDB Integration

T2.1.1: Test updateData ()

function updateData(process,data,setlistID)

switch setlistID

case 'AmountLimit'

process.InjectorDB.SetAmountLimit(data);

case 'AmountInShortPeriod'

process.InjectorDB.SetAmountInShortPeriod(data);

case 'Baseline'

process.InjectorDB.SetBaseline(data);

case 'Bolus'

process.InjectorDB.SetBolus(data);

end

end

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T2.1.1.1 |
| Coverage Item | Tcover2.1.1.1 |
| Input | data = 1;  setlistID1 = 'AmountLimit';  setlistID2 = 'AmountInShortPeriod';  setlistID3 = 'Baseline';  setlistID4 = 'Bolus'; |
| State | db=InjectorDB; |
| Expected Output | db.AmountLimit=1  db.AmountInShortPeriod=1  db.Baseline=1  db.Bolus=1 |

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

T2.1.2: Test updateAuthority ()

function updateAuthority(process,bol)

process.InjectorDB.SetAuthority(bol);

end

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T2.1.2.1 |
| Coverage Item | Tcover2.1.2.1 |
| Input | data = 1;  pro.updateAuthority(data); |
| State | db=InjectorDB; |
| Expected Output | db.Authority=1 |

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

T2.1.3: Test getData ()

function data = getdata(process,id)

switch id

case 'Baseline'

data = process.InjectorDB.Baseline;

case 'Bolus'

data = process.InjectorDB.Bolus;

case 'BaselineforEm'

data = process.InjectorDB.BaselineforEm;

case 'BolusforEm'

data = process.InjectorDB.BolusforEm;

end

end

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T2.1.3.1 |
| Coverage Item | Tcover2.1.3.1 |
| Input | data = 'empty';  setlistID1 = 'Baseline';  setlistID2 = 'Bolus';  setlistID3 = 'BaselineforEm';  setlistID4 = 'Bolus'; |
| State | db=InjectorDB; |
| Expected Output | getdata(setlistID1)=data  getdata(setlistID2)=data  getdata(setlistID3)=data  getdata(setlistID4)=data |

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

### T2.2: InjectorUI+InjectorProcessor+InjectorDB Integration

T2.2.1: Test input

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T2.2.1.1 |
| Coverage Item | Tcover2.2.1.1 |
| Input | input Baseline = ‘0.1’  input Bolus = ‘0.5’  input BaselineforEm = ‘0.1’  input BolusForEm = ‘0.2’ |
| State | initial state |
| Expected Output | db.Baseline=’0.1’  db.Bolus=’0.5’  db.BaselineforEm = ‘0.1’  db.BolusForEm = ‘0.2’ |

|  |  |
| --- | --- |
|  | Test Case T2.2.1.2 |
| Coverage Item | Tcover2.2.1.2 |
| Input | input Baseline = ‘0.2’  input Bolus = ‘0.5’  input BaselineforEm = ‘0.1’  input BolusForEm = ‘0.2’ |
| State | initial state |
| Expected Output | db.Baseline=’0’  db.Bolus=’0.5’  db.BaselineforEm = ‘0.1’  db.BolusForEm = ‘0.2’ |

|  |  |
| --- | --- |
|  | Test Case T2.2.1.3 |
| Coverage Item | Tcover2.2.1.3 |
| Input | input Baseline = ‘0.1’  input Bolus = ‘0.6’  input BaselineforEm = ‘0.1’  input BolusForEm = ‘0.2’ |
| State | initial state |
| Expected Output | db.Baseline=’0.1’  db.Bolus=’0’  db.BaselineforEm = ‘0.1’  db.BolusForEm = ‘0.2’ |

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

## T3: Functional Test

T3.1: Use Case “emergency shot”

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T3.1.1 |
| Coverage Item | Tcover3.1.1 |
| Input | testCase.press(testCase.app.powerSwitch);  testCase.press(testCase.app.AuthoritySwitch); testCase.type(testCase.app.inputEditField,'0.05');  testCase.choose(testCase.app.setlistDropDown,'Bolus');  testCase.type(testCase.app.inputEditField,'0.2');  testCase.choose(testCase.app.setlistDropDown,'BolusforEm');  testCase.type(testCase.app.inputEditField,'0.2');  testCase.choose(testCase.app.setlistDropDown,'BaselineforEm');  testCase.type(testCase.app.inputEditField,'0.05'); |
| State | testCase.app=InjectorUI;  testCase.app2=Emergency; |
| Expected Output | the baseline increase when the patient pushes the emergency shot button. |

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

T3.2: Situation “Patient push emergency during injection (without authority)”

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T3.2.1 |
| Coverage Item | Tcover3.2.1 |
| Input | testCase.press(testCase.app.powerSwitch); testCase.type(testCase.app.inputEditField,'0.05');  testCase.choose(testCase.app.setlistDropDown,'Bolus');  testCase.type(testCase.app.inputEditField,'0.2');  testCase.choose(testCase.app.setlistDropDown,'BolusforEm');  testCase.type(testCase.app.inputEditField,'0.2');  testCase.choose(testCase.app.setlistDropDown,'BaselineforEm');  testCase.type(testCase.app.inputEditField,'0.05'); |
| State | testCase.app=InjectorUI;  testCase.app2=Emergency; |
| Expected Output | The patient can’t use the emergency shot button. |

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

T3.3: Situation “The doctor inputs illegal baseline and bolus”

* Coverage Criteria: branch coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T3.3.1 |
| Coverage Item | Tcover3.3.1 |
| Input | testCase.press(testCase.app.powerSwitch); testCase.type(testCase.app.inputEditField,'0.5'); |
| State | testCase.app=InjectorUI;  testCase.app2=Emergency; |
| Expected Output | There is an error information for illegal baseline |

|  |  |
| --- | --- |
|  | Test Case T3.3.2 |
| Coverage Item | Tcover3.3.2 |
| Input | testCase.press(testCase.app.powerSwitch);  testCase.choose(testCase.app.setlistDropDown,'Bolus'); testCase.type(testCase.app.inputEditField,'0.2'); |
| State | testCase.app=InjectorUI;  testCase.app2=Emergency; |
| Expected Output | There is an error information for illegal bolus |

* Test coverage: 2/2=100%
* Test result: 2 passed

T3.4: Situation “The amount reaches the amount limit for one hour”

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T3.4.1 |
| Coverage Item | Tcover3.3.1 |
| Input | testCase.type(testCase.app.inputEditField,'0.1');  testCase.choose(testCase.app.setlistDropDown,'Bolus');  testCase.type(testCase.app.inputEditField,'0.5');  testCase.choose(testCase.app.setlistDropDown,'BolusforEm');  testCase.type(testCase.app.inputEditField,'0.2');  testCase.choose(testCase.app.setlistDropDown,'BaselineforEm');  testCase.type(testCase.app.inputEditField,'0.05');  testCase.press(testCase.app.startButton);  testCase.choose(testCase.app.setlistDropDown,'Baseline');  testCase.type(testCase.app.inputEditField,'0.1');  testCase.choose(testCase.app.setlistDropDown,'Bolus');  testCase.type(testCase.app.inputEditField,'0.3');  testCase.choose(testCase.app.setlistDropDown,'BolusforEm');  testCase.type(testCase.app.inputEditField,'0.2');  testCase.choose(testCase.app.setlistDropDown,'BaselineforEm');  testCase.type(testCase.app.inputEditField,'0.05');  testCase.press(testCase.app.startButton);  testCase.type(testCase.app.inputEditField,'0.1');  testCase.choose(testCase.app.setlistDropDown,'Bolus');  testCase.type(testCase.app.inputEditField,'0.5');  testCase.choose(testCase.app.setlistDropDown,'BolusforEm');  testCase.type(testCase.app.inputEditField,'0.2');  testCase.choose(testCase.app.setlistDropDown,'BaselineforEm');  testCase.type(testCase.app.inputEditField,'0.05');  testCase.press(testCase.app.startButton); |
| State | testCase.app=InjectorUI;  testCase.app2=Emergency; |
| Expected Output | The injector pauses at the third time. The db.TotalAmount equals to 1. |

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

## T4: Modal checking (Injector.xml)

The whole system could be implemented by two models (CalculateBolus and CalculateHour). CalculateBolus implements the rule to control the baseline during the injection. CaculateHour implement how the system make the amount of injectant is less than 1ml. Relation between two models: the inject\_state in CalculateHour could be replace by CalculateBolus. (The implementations in UPPAAL and the implementations in MATLAB are same)

T4.1 CalculateBolus(UPPAAL)

地图的截图

描述已自动生成

The start state is the initial state, when doctor push start button, the system will enter the main\_inject state. And if the patient pushes the emergency button, the system will enter the main\_emergency state. As for main injection could finish earlier than emergency shot, the system could enter the emergency\_only state.For each injection, it should end at finish\_state.

T4.2 CaculateHour(UPPAAL)

图片包含 游戏机, 文字, 地图

描述已自动生成

The start state is the initial state, when doctor push start button, the system will enter the inject\_state. If the total amount reaches the amount limit for one hour, the system will enter the stop\_state, and exit the stop\_state when the sum of items in oneHourCache(which holds the amount from one hour ago to current time) less than amount limit for one hour. And when one injection finish, the system would enter the finish\_state.