Group 14

Author: Chen Fang

Validation Report

Proj.3 Painkiller Injection System

Table of Contents

[System Architecture 2](#_Toc44274194)

[T1: Unit Test 2](#_Toc44274195)

[T1.1: InjectionDB Unit Test 2](#_Toc44274196)

[T2: Integration Test 3](#_Toc44274197)

[T2.1: InjectionProcessor+PhysicianUI Integration 3](#_Toc44274198)

[T2.2: InjectionProcessor+PhysicianUI+PatientUI Integration 4](#_Toc44274199)

[T3: Functional Test 5](#_Toc44274200)

[T3.1: Use Case “Physician exchange the painkiller amount of Bolus injection ”&“Patient make request for Bolus injection” 5](#_Toc44274201)

[T3.2: Use Case “Physician exchange the painkiller amount of Baseline injection” & “Physician give a Baseline injection, while the amount of painkiller updated as time passed.” 6](#_Toc44274202)

[T3.3: Use Case “Physician give a Baseline injection and stop the process manually.” 6](#_Toc44274203)

[Model Checking 7](#_Toc44274204)

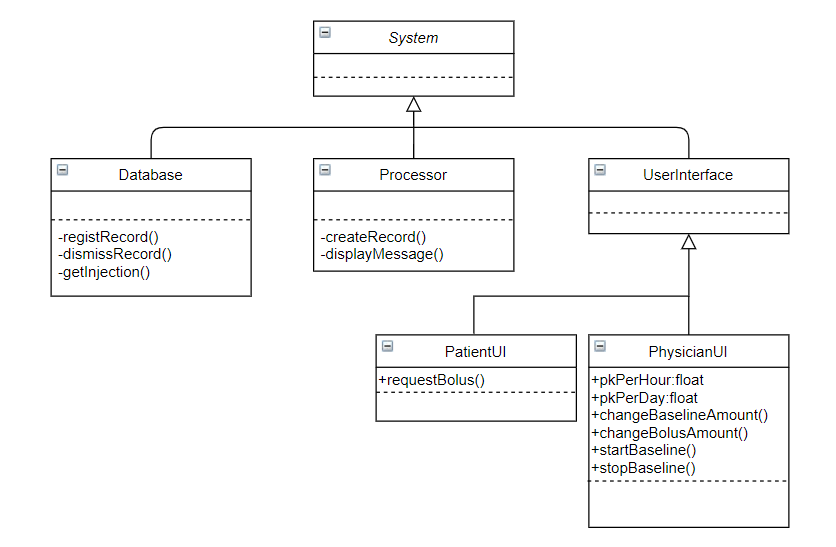
[Patient model 7](#_Toc44274205)

[Physician model 7](#_Toc44274206)

[Updating system model 8](#_Toc44274207)

## System Architecture

The system architecture is shown below:



## T1: Unit Test

### T1.1: InjectionDB Unit Test

T1.1.1: Test regist()

function regist(db,record)

db.recordingList=[db.recordingList;record]; Tcover1.1.1.1

end

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.1.1.1 |
| Coverage Item | Tcover1.1.1.1 |
| Input | re=PKrecord;  re.way='Bolus';  re.painkillerAmount=0.2;  re.startTime=’12:00’;  re.finishTime=’12:00’; |
| State | db=InjectionDB; |
| Expected Output | re==db.recordingList(1) |

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

T1.1.2: Test dismissRecord()

function dismissRecord(db)

db.recordingList=[]; Tcover1.1.2.1

end

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.1.2.1 |
| Coverage Item | Tcover1.1.2.1 |
| Input | re=PKrecord;  re.way='Bolus';  re.painkillerAmount=0.2;  re.startTime=’12:00’;  re.finishTime=’12:00’; |
| State | db=InjectionDB;  db.regist(re); |
| Expected Output | db.recordingList==[] |

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

## T2: Integration Test

### T2.1: InjectionProcessor+PhysicianUI Integration

T2.1.1: Test createRecord(way, PKamount)

function [record,result] = createRecord(processor,way,pkAmount)

record.way = way;

record.painkillerAmount = pkAmount;

if processor.physicianApp.time < 10 && processor.physicianApp.count < 10

record.startTime = sprintf('0%d:0%d',processor.physicianApp.time,processor.physicianApp.count);

elseif processor.physicianApp.time >= 10 && processor.physicianApp.count < 10

record.startTime = sprintf('%d:0%d',processor.physicianApp.time,processor.physicianApp.count);

elseif processor.physicianApp.time < 10 && processor.physicianApp.count >= 10

record.startTime = sprintf('0%d:%d',processor.physicianApp.time,processor.physicianApp.count);

else

record.startTime = sprintf('%d:%d',processor.physicianApp.time,processor.physicianApp.count);

end

result = 'Fail';

switch way

case 'Baseline'

if processor.physicianApp.PKperday < 3 && processor.physicianApp.PKperhour < 1

result = 'Success';

record.finishTime = '';

end

case 'Bolus'

if processor.physicianApp.PKperday < 3 && 1 - processor.physicianApp.PKperhour >= pkAmount

result = 'Success';

record.finishTime = '';

end

end

end

* Coverage Criteria: Branch coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T2.1.1.1 |
| Coverage Item | Tcover2.1.1.1 |
| Input | ph = PhysicianUI;  pro = InjectionProcessor;  pro.physicianApp = ph;  ph.Pro = pro; |
| Branch | [record,result] = pro.createRecord('Baseline',0.04); |
| Expected Output | 'Baseline' == record.way;  0.04 == record.painkillerAmount;  'Success' == result; |

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

|  |  |
| --- | --- |
|  | Test Case T2.1.1.2 |
| Coverage Item | Tcover2.1.1.2 |
| Input | ph = PhysicianUI;  pro = InjectionProcessor;  pro.physicianApp = ph;  ph.Pro = pro; |
| Branch | [record,result] = pro.createRecord('Bolus',0.2); |
| Expected Output | 'Bolus' == record.way;  0.2 == record.painkillerAmount;  'Success' == result; |

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

### T2.2: InjectionProcessor+PhysicianUI+PatientUI Integration

T2.2.1 Test displayMessage(mes,who)

function displayMessage(processor,mes,who)

switch who

case 'Physician'

processor.physicianApp.Message.Text = [processor.physicianApp.Message.Text;{mes}];

case 'Patient'

processor.patientApp.Message.Text = [processor.patientApp.Message.Text;{mes}];

end

end

* Coverage Criteria: Branch coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T2.2.1.1 |
| Coverage Item | Tcover2.2.1.1 |
| Input | ph = PhysicianUI;  pa = PatientUI;  pro = InjectionProcessor;  pro.physicianApp = ph;  pro.patientApp = pa;  ph.Pro = pro;  ph.patient = pa;  pa.Pro = pro;  pa.physician = ph; |
| Branch | pro.displayMessage('Hello!','Physician'); |
| Expected Output | {'Hello!'} == ph.Message.Text(end); |

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

|  |  |
| --- | --- |
|  | Test Case T2.2.1.2 |
| Coverage Item | Tcover2.2.1.2 |
| Input | ph = PhysicianUI;  pa = PatientUI;  pro = InjectionProcessor;  pro.physicianApp = ph;  pro.patientApp = pa;  ph.Pro = pro;  ph.patient = pa;  pa.Pro = pro;  pa.physician = ph; |
| Branch | pro.displayMessage('Hello!','Patient'); |
| Expected Output | {'Hello!'} == pa.Message.Text(end); |

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

## T3: Functional Test

### T3.1: Use Case “Physician exchange the painkiller amount of Bolus injection ”&“Patient make request for Bolus injection”

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T3.1.1.1 |
| Coverage Item | Tcover3.1.1.1 |
| Input | testCase.type(testCase.Physician.BolusamountEditField,0.5);  testCase.press(testCase.Physician.ChangeButton);  testCase.press(testCase.Patient.RequestButton);  testCase.press(testCase.Patient.RequestButton);  testCase.press(testCase.Patient.RequestButton); |
| State | testCase.Patient = PatientUI;  testCase.Physician = PhysicianUI;    Database = InjectionDB;  Processor = InjectionProcessor; |
| Expected Output | Physician.BoluPK == 0.5  Physician.PKperhour == 1  Physician.RecordingList.Data{1} == ‘Bolus’ |

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

### T3.2: Use Case “Physician exchange the painkiller amount of Baseline injection” & “Physician give a Baseline injection, while the amount of painkiller updated as time passed.”

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T3.2.1.1 |
| Coverage Item | Tcover3.2.1.1 |
| Input | testCase.type(testCase.Physician.BaselineamountEditField,0.05);  testCase.press(testCase.Physician.ConfirmButton);  testCase.press(testCase.Physician.StartButton);  testCase.press(testCase.Physician.StartButton); |
| State | testCase.Patient = PatientUI;  testCase.Physician = PhysicianUI;    Database = InjectionDB;  Processor = InjectionProcessor; |
| Expected Output | Physician.BasePK == 0.05  Physician.RecordingList.Data{1} == ‘Baseline’  Physician.PKperhour == 0  Physician.PKperday == 0 |

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

NOTES: To accelerate the process, use the other day timer in *PhysicianUI.*

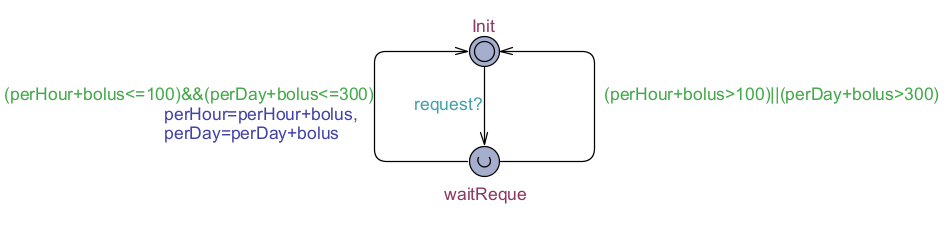
### T3.3: Use Case “Physician give a Baseline injection and stop the process manually.”

To verify the function of stop Baseline injection manually, you can first instantiate the *InjectionProcessor, InjectionDB* and *PhysicianUI* (and maybe *PatientUI*, but it is not essential for this use case). Then you first set the painkiller amount for Baseline injection by change the value in *Baseline Amount* edit field and push *Confirm* button, and push *Start* button to start injection. Before reaching the limit of 1ml/hour, you can push *Stop* button to stop the process at any time you want. The *Amount/Hour* and *Amount/Day* will be updated with the process and you can verify the stop function by watching the value and see whether the task has been completed in the *RecordingList* panel and *Message* panel on *PhysicianUI*.

NOTES:As the multitask problem in matlab, we can not give a test case like use case above to validate the function.

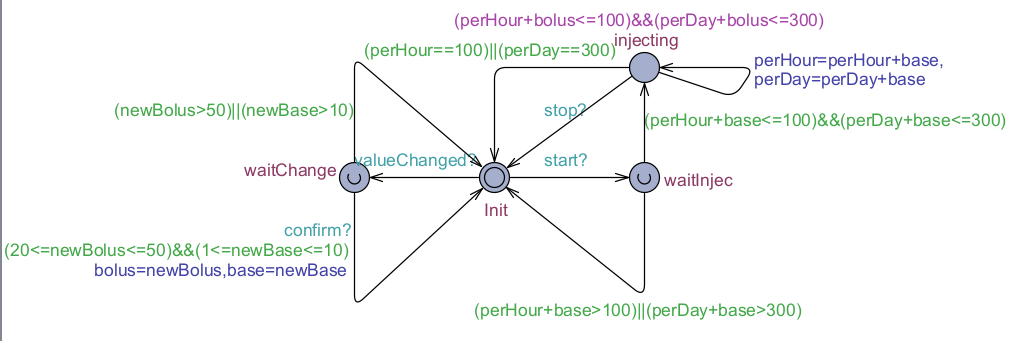
## Model Checking

### Patient model



When get a *request* signal which means the patient push the *Request* button on the patient UI, the system would judge whether the patient could get a bolus injection. If he/she does not reach the limit, then the bolus injection would work and update the information of painkiller/hour and painkiller/day.

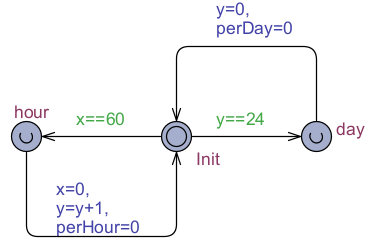
### Physician model



When get a *valuechanged* signal which means the value of *Bolus Amount* or *Baseline Amount* on the physician UI has been changed, the system would first check whether the input is a valid one and then update the corresponding value.

When get a *start* signal which means the *Start* button on the physician UI has been pushed, the system would first check whether the patient could at least get a one-minute injection. If he/she does not reach the limit, the injection would start and would stop until reach the limit or get a *stop* signal which means the *Stop* button on the physician UI has been pushed and the process needs to stop.

### Updating system model



The system uses a clock and an integer to simulate the time. When *clock x* equals to 60 which means one hour has passed, the system would increase *integer y* and update the *perHour* representing the update of information on the physician UI. Similarly, as *integer y* represents the hour, when it equals to 24, one day has passed and the perDay would update.

NOTES: As UPPAAL cannot represent float numbers, we scaled the numbers (bolus, base, perHour, perDay, etc.) by 100 to make them integers.