SOFTWARE SPECIFICATIONS

Painkiller System

Group 16
Author: 王书悦

Table of Contents

System Architecture

The system architecture is shown below:

Since MatLab is not able to test private functions, so I change all private functions to public functions in a new folder for the convenience of testing.

T1: Unit Test

T1.1: SystemProcessor Unit Test

T1.1.1: Test updateInjectionSpeed ()

```
function injectionSpeed = updateInjectionSpeed(pro, time)
      % Check if normal injection should be on or off.
      % Also update the database about the decision.
      %   Input:  time [datetime]
      %             current time
      %   Return: injectionSpeed [float]
      %             the speed of the injection
      db = pro.sysDB;

      % Check if the switch is on.
      if pro.isOn && db.getPainkillerRemaining(pro.time) > 0
      % Branch – Tcover1.1.1.1
          injectionSpeed = db.getBaseline();
      else  % Branch – Tcover1.1.1.2
          injectionSpeed = 0;
      end

      % Check 1 hour limit
      last1h = db.getPainkillerLast1h(time);

      % Check 24 hour limit
      last24h = db.getPainkillerLast24h(time);

      if (last24h >= db.getDailyLimit()) || (last1h >=
            db.getHourlyLimit())    % Branch – Tcover1.1.1.3
          injectionSpeed = 0;
      end

      db.updateInjectionSpeed(time, injectionSpeed);
  end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.1.1.1	Test Case T1.1.1.2	Test Case T1.1.1.3
Coverage Item	Tcover1.1.1.1	Tcover1.1.1.2	Tcover1.1.1.3
Input	time = pro.time;	time = pro.time;	time = pro.time;

State	pro = SystemProcessor;
	pro.isOn = 0;
		pro = SystemProcessor;
		db.baseline = 0.02;
		pro.isOn = 1;
			db = SystemDB
			db.baseline = 0.1;
			pro.speed = 3600;
			pro.isOn = 1; pause(1);

Expected Output	injectionSpeed==0	injectionSpeed==0.02	injectionSpeed==0

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

T1.1.2: Test refresh()

```
        function refresh(pro,~,~)
            % The timer update function. This function will be called every
            % 0.1 second once functino 'run' is called.
            % This function will be invisible from outside, and will never
            % be called by objects except proTimer.

            db = pro.sysDB;

            % Update current time.
            currectTime = datetime();
            pro.time = pro.time + pro.speed*(currectTime - pro.tickTimeTag);
            pro.tickTimeTag = currectTime;

            % Update current injection speed.
            injectionSpeed = pro.updateInjectionSpeed(pro.time);

            % Gathering information...
            % Info for last 1 h
            last1h = db.getPainkillerLast1h(pro.time);
            limit1h = db.getHourlyLimit();
            last1h = sprintf('%0.2fml / %0.2fml',min([last1h, limit1h]),
limit1h);

            % Info for last 24 h
            last24h = db.getPainkillerLast24h(pro.time);
            limit24h = db.getDailyLimit();
            last24h = sprintf('%0.2fml / %0.2fml',min([last24h, limit24h]),
limit24h);

            % Info about bolus
            possibleBolus = db.getPossibleBolus(pro.time);
            interval = db.getInterval();
            lastTime = db.getLastInjectionTime();

            % Info about painkiller left
            painkillerLeft = db.getPainkillerRemaining(pro.time);

            % Show the result
            pro.mainApp.show(pro.time, injectionSpeed, last1h, last24h,...
                possibleBolus, interval, lastTime, painkillerLeft);
            % Control the bolus button
            if possibleBolus == 0 || ~pro.isOn  % Branch – Tcover1.1.2.1
                pro.bolusApp.BolusButton.Enable = 'off';
            else  % Branch – Tcover1.1.2.2
                pro.bolusApp.BolusButton.Enable = 'on';
            end
        end
    end
```

- Coverage Criteria: Branch coverage

- Test case

	Test Case T1.1.2.1	Test Case T1.1.2.2
Coverage Item	Tcover1.1.2.1	Tcover1.1.2.2
Input	-----	-----
State	Pro = SystemProcessor; pro.isOn = 0;	Pro = SystemProcessor; pro.isOn = 1;
Expected Output	injectionSpeed==0	injectionSpeed==0.02

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

T1.1.3: Test getCurTime()

```
function time = getCurTime(pro)
    % Get the current time tag in processor.
    %   Input:  None
    %   Return: time [datetime]
    %             current time in processor.
    time = pro.time; % Statement - Tcover1.1.3.1
end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.1.3.1
Coverage Item	Tcover1.1.3.1
Input	-----

State	Pro = SystemProcessor;
	y1 = datetime();pause(1);
	x = pro.getCurTime();pause(1);
	y2 = datetime();

Expected Output	Time is between y1 and y2

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

T1.1.4: Test getCurSpeed ()

```
function speed = getCurSpeed(pro)
      % Get the current time passing speed in processor.
      %   Input:  None
      %   Return: speed [int]
      %             current time passing speed in processor.
      speed = pro.speed; % Statement - Tcover1.1.4.1
end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.1.4.1
Coverage Item	Tcover1.1.4.1
Input	-----

State	Pro = SystemProcessor; pro.speed = 3600;
Expected Output	3600

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

T1.1.5: Test setSpeed()

```
function setSpeed(pro, barValue)
    % Set the current time passing speed in processor.
    %   Caller: MainBoardUI.TimeElapseSpeedSlider - callback
    %           TimeElapseSpeedSliderValueChanged(event)
    %   Input:  barValue [float]
    %             bar value read from the slider of the UI.
    %             value should scaled from 0 to 2.
    %             speed will be log scaled from 1x to 3600x. Always
    %             int.
    %   Return: None
    pro.speed = floor(60^barValue); % Statement - Tcover1.1.5.1
end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.1.5.1
Coverage Item	Tcover1.1.5.1
Input	2
State	Pro = SystemProcessor;
Expected Output	pro.speed = 3600

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

T1.1.6: Test setSwitch()

```
function setSwitch(pro, switchValue)
    % Set the switch to be consistent with the machine switch.
    %   Caller: MainBoardUI.Switch - callback
    %           SwitchValueChanged(event)
    %   Input:  switchValue [string]
    %             'On' is the machine is on, 'Off' if the machine
    %             is off.
    %   Return: None
    if strcmp(switchValue, 'On')  % Branch – Tcover1.1.6.1
        pro.isOn = 1;
    else  % Branch – Tcover1.1.6.2
        pro.isOn = 0;
    end
end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.1.6.1	Test Case T1.1.6.2
Coverage Item	Tcover1.1.6.1	Tcover1.1.6.2
Input	‘On’	‘Off’
State	Pro = SystemProcessor;	Pro = SystemProcessor;

Expected Output	pro.isOn == 1	pro.isOn == 0

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

T1.1.7: Test setPainkillerFull ()

```
function setPainkillerFull(pro)
    % Add the painkiller to full.
    %   Caller: MainBoardUI.AddPainkillerButton - callback
    %           AddPainkillerButtonPushed(event)
    %   Input:  None
    %   Return: None
    pro.sysDB.addPainkiller(pro.time,10); % Statement - Tcover1.1.7.1
end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.1.7.1
Coverage Item	Tcover1.1.7.1
Input	-----
State	Pro = SystemProcessor; db = SystemDB;
Expected Output	Db. painkillerRemaining = 10

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

T1.1.8: Test bolusPressed()

```
function bolusPressed(pro)
    % Try to make a bolus.
    %   Caller: BolusUI.BolusButton - callback
    %           BolusButtonPushed(event)
    %   Input:  None
    %   Return: None
    db = pro.sysDB;
    if round(db.getPossibleBolus(pro.time),2) ~= 0 && pro.isOn
        db.makeBolus(pro.time); % Branch - Tcover1.1.8.1
    end
end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.1.8.1	Test Case T1.1.8.2
Coverage Item	Tcover1.1.8.1	Tcover1.1.8.1
Input	-----	------
State	Pro = SystemProcessor; db = SystemDB; tc.pro.isOn = 1;	Pro = SystemProcessor; db = SystemDB; tc.pro.isOn = 1;
Expected Output	db.bolusDataList(end).amount == tc.db.bolus	db.bolusDataList is empty

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

T1.2: SystemDB Unit Test

T1.2.1: Test createNormalData()

```
function createNormalData(DB, time, speed)
        % Add a data line in normal data.
        % Only accessible inside database.
        %   Caller: SystemDB.updateInjectionSpeed(time, injectionSpeed)
        %   Input:  time [datatime]
        %             current time of processor. used as start time
        %             point of inserted injection data.
        %           speed [float]
        %             injection speed of the inserted data.
        %   Return: None
        nd = NormalData;  % Statement - Tcover1.2.1.1
        nd.startTime = time;
        nd.speed = speed;
        nd.endTime = DB.DUMMYTIME;
        DB.normalDataList = [DB.normalDataList; nd];
    end
end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.2.1.1
Coverage Item	Tcover1.2.1.1
Input	datetime(), 1
State	Pro = SystemProcessor; db = SystemDB;
Expected Output	db. normalDataList (end).startTime == datetime(); db. normalDataList (end).speed == 1;

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

T1.2.2: Test getPainkillerBetween()

```
    function amount = getPainkillerBetween(DB, startTime, endTime)
        % Get the amount of painkiller injected
        % between start time and end time.
        %   Input:  startTime [datetime]
        %             start time point of the query.
        %           endTime [datetime]
        %             end time point of the query.
        %   Return: amount [float]
        %             total injection (include normal and bolus) made
        %             between start time point and end time point.
        amount = 0;

        % Check bolus data
        for data = DB.bolusDataList'
            if startTime < data.injectTime && data.injectTime <= endTime
                amount = amount + data.amount;
              % Branch - Tcover1.2.2.1
            end
        end
```

```
            % Check normal data
            for data = DB.normalDataList'
                if data.endTime == DB.DUMMYTIME && endTime > data.startTime
                    du = endTime - max(startTime, data.startTime);
                  % Branch - Tcover1.2.2.2
                    amount = amount + data.speed * minutes(du);
                elseif data.endTime > startTime && data.startTime < endTime
                  % Branch - Tcover1.2.2.3
                    du = min(endTime, data.endTime) - max(startTime,
                  data.startTime);
                    amount = amount + data.speed * minutes(du);
                end
            end
        end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.2.2.1
Coverage Item	Tcover1.2.2.1, Tcover1.2.2.2, Tcover1.2.2.3
Input	pro.time -duration(24,0,0), pro.time
State	Pro = SystemProcessor; db = SystemDB; Make the system on, let time speed to 3600x, press bolusButton, pause 5 seconds;
Expected Output	3

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

T1.2.3: Test getRecentBolusDataList()

```
        function bolusDataList = getRecentBolusDataList(DB)
            % Get the most recent 10 records in bolus data list.
            %   Input:  None
            %   Return: bolusDataList [array[string](size=[[0,10], 2])]
            %             most recent 10 records in bolus data list.
            %             each row contains 2 data, first is the injection
            %             time, second is the injection amount.
            %             all data in string format.
            bolusDataList = repmat(["NULL",
0],min(10,length(DB.bolusDataList)),1); % Statement - Tcover1.2.3.1
            for i=1:min(10,length(DB.bolusDataList))
                bolusDataList(i,1) =
string(datetime(DB.bolusDataList(length(DB.bolusDataList)+1-
i).injectTime,'Format','yyyy-MM-dd HH:mm:ss'));
                bolusDataList(i,2) =
round(DB.bolusDataList(length(DB.bolusDataList)+1-i).amount, 2);
            end
        end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.2.3.1
Coverage Item	Tcover1.2.3.1

Input	------
State	Pro = SystemProcessor; db = SystemDB; Make the system on, press bolusButton;
Expected Output	bolusDataList(1,2) == 0.3

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

T1.2.4: Test getRecentNormalDataList()

```
        function normalDataList = getRecentNormalDataList(DB)
            % Get the most recent 10 records in normal data list.
            %   Input:  None
            %   Return: normalDataList [array[string](size=[[0,10], 3])]
            %             most recent 10 records in bolus data list.
            %             each row contains 3 data, first is the starting
            %             injection time, second is the ending injection
            %             time, third is the injection speed.
            %             all data in string format.
            normalDataList =
repmat(["start","end",0],min(10,length(DB.normalDataList)),1);
            for i=1:min(10,length(DB.normalDataList))
                normalDataList(i,1) =
                  string(datetime(DB.normalDataList(length(DB.normalDataList)
                  +1-i).startTime,'Format','yyyy-MM-dd HH:mm:ss'));
                if DB.normalDataList(length(DB.normalDataList)+1-i).endTime
                  ~= DB.DUMMYTIME % Branch - Tcover1.2.4.1
                    normalDataList(i,2) =
                        string(datetime(DB.normalDataList(length(DB.normalDat
                  aList)+1-i).endTime,'Format','yyyy-MM-dd HH:mm:ss'));
                else    % Branch - Tcover1.2.4.2
                    normalDataList(i,2) = "--(now)--";
                end
                normalDataList(i,3) =
round(DB.normalDataList(length(DB.normalDataList)+1-i).speed, 2);
            end
        end
    end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.2.4.1
Coverage Item	Tcover1.2.4.1, Tcover1.2.4.2
Input	------
State	Pro = SystemProcessor; db = SystemDB; Set baseline = 0.2, speed = 60x, make the system on, pause 1 second; Set baseline = 0.2, pause 1 second;
Expected Output	NormalDataList (2,3) == 0.2, NormalDataList (1,2) == "--(now)--"

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

T1.2.5: Test getPainkillerLast1h()

```
function amount = getPainkillerLast1h(DB, time)
    % Get the amount of painkiller injected within 1 hour.
    % Call public function SystemDB.getPainkillerBetween().
    %   Input:  time [datetime]
    %             current time in processor.
    %   Return: amount [float]
    %             the amount of painkiller from 1 hour ago to
    %             current time point.
    amount = DB.getPainkillerBetween(time-duration(1,0,0), time);
    % Statement - Tcover1.2.5.1
end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.2.5.1
Coverage Item	Tcover1.2.5.1
Input	detetime()
State	Pro = SystemProcessor; db = SystemDB; Set baseline = 0, make a bolus;
Expected Output	amount == db.bolus

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

T1.2.6: Test getPainkillerLast24h()

```
function amount = getPainkillerLast24h(DB, time)
    % Get the amount of painkiller injected within 24 hours.
    % Call public function SystemDB.getPainkillerBetween().
    %   Input:  time [datetime]
    %             current time in processor.
    %   Return: amount [float]
    %             the amount of painkiller from 24 hours ago to
    %             current time point.
    amount = DB.getPainkillerBetween(time-duration(24,0,0), time);
    % Statement - Tcover1.2.6.1
end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.2.6.1
Coverage Item	Tcover1.2.6.1
Input	detetime()
State	Pro = SystemProcessor; db = SystemDB; Set baseline = 0, make a bolus;
Expected Output	amount == db.bolus

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

T1.2.7: Test getPainkillerRemaining()

```
function amount = getPainkillerRemaining(DB, time)
    % Get the remaining painkiller in the system.
    %   Input:  time [datetime]
    %             current time in processor
    %   Return: amount [float]
    %             the amount of painkiller remaining in the system.
    amount = DB.painkillerRemaining -...
        DB.getPainkillerBetween(DB.painkillerCheckpoint, time);
    % Statement - Tcover1.2.7.1
    amount = max([amount, 0]);
end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.2.7.1
Coverage Item	Tcover1.2.7.1
Input	detetime()
State	Pro = SystemProcessor; db = SystemDB; Set baseline = 0, add painkiller to full (10 ml), make a bolus;
Expected Output	amount == 10 - db.bolus

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

T1.2.8: Test getPossibleBolus()

```
function amount = getPossibleBolus(DB, time)
    % Return the amount of the possible maximum bolus currently.
    %   Input:  time [datetime]
    %             current time in processor
    %   Return: amount [float]
    %             the maximum amount of painkiller possible if make
    %             a bolus at this time.
    last1h = DB.getPainkillerLast1h(time);
    last24h = DB.getPainkillerLast24h(time);
    amount = max([0, min([...
                    DB.bolus,...
                    DB.hourlyLimit - last1h,...
                    DB.dailyLimit - last24h,...
                    DB.getPainkillerRemaining(time)])]);
    if ~isempty(DB.bolusDataList) &&...
            DB.bolusDataList(end).injectTime + ...
            duration(0, DB.interval, 0) > time
    % Branch - Tcover1.2.8.1
        amount = 0;
    end

end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.2.8.1	Test Case T1.2.8.2

Coverage Item	Tcover1.2.8.1	Tcover1.2.8.1
Input	detetime()	detetime()
State	Pro = SystemProcessor; db = SystemDB; Set baseline = 0, turn on the system;	Pro = SystemProcessor; db = SystemDB; Set baseline = 0, turn on the system, make a bolus;
Expected Output	amount == db.bolus	amount == 0

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

T1.2.9: Test getLastInjectionTime()

```
        function lastTime = getLastInjectionTime(DB)
            % Return a string representing the last time injecting a bolus.
            %   Input:  None
            %   Return: lastTime [string]
            %             string representing the last time injecting a
            %             bolus. Format should be 'yyyy-MM-dd HH:mm'. If no
            %             bolus has been made, return  'No record...'.
            if isempty(DB.bolusDataList) % Branch - Tcover1.2.9.1
                lastTime = 'No record...';
            else  % Branch - Tcover1.2.9.2
                lastTime = DB.bolusDataList(end).injectTime;
                lastTime = string(datetime(lastTime,'Format','yyyy-MM-dd
HH:mm'));
            end
        end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.2.9.1	Test Case T1.2.9.2
Coverage Item	Tcover1.2.9.1	Tcover1.2.9.2
Input	-----	-----
State	Pro = SystemProcessor; db = SystemDB; Set baseline = 0, turn on the system;	Pro = SystemProcessor; db = SystemDB; Set baseline = 0, turn on the system, make a bolus;
Expected Output	lastTime == 'No record...'	lastTime ~= 'No record...'

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

T1.2.10: Test checkPassword()

```
function passed = checkPassword(DB, password)
    % Check the correctness of the input password.
    %   Input:  password [string]
    %             the password user inputs, waiting to check.
    %   Return: passed [logical]
    %             true if the password is correct. Otherwise false.
    if strcmp(DB.password, password) % Branch - Tcover1.2.10.1
        passed = true;
    else  % Branch - Tcover1.2.10.2
        passed = false;
    end
end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.2.10.1	Test Case T1.2.10.2
Coverage Item	Tcover1.2.10.1	Tcover1.2.10.2
Input	DB.password	123
State	Pro = SystemProcessor; db = SystemDB;	Pro = SystemProcessor; db = SystemDB;
Expected Output	true	false

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

T1.2.11: Test getParameters()

```
function [baseline,dailyLimit,...
        hourlyLimit,bolus,...
        interval,password] = getParameters(DB, password)
    % Get all the parameters from database, used for setting.
    %   Caller: SettingUI.startupFcn()
    %   Input:  password [string]
    %             the password user inputs, waiting to check.
    %   Return: baseline [float]
    %           dailyLimit [float]
    %           hourlyLimit [float]
    %           bolus [float]
    %           interval [float]
    %           password [string]
    %             parameters from database.
    passed = DB.checkPassword(password);
    if passed   % Branch - Tcover1.2.11.1
        baseline    = DB.baseline;
        dailyLimit  = DB.dailyLimit;
        hourlyLimit = DB.hourlyLimit;
        bolus       = DB.bolus;
        interval    = DB.interval;
        password    = DB.password;
    else        % Branch - Tcover1.2.11.2
        baseline    = 0;
```

```
            dailyLimit  = 0;
            hourlyLimit = 0;
            bolus       = 0;
            interval    = 0;
            password    = 'Invalid password!';
        end
    end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.2.11.1	Test Case T1.2.11.2
Coverage Item	Tcover1.2.11.1	Tcover1.2.11.2
Input	DB.password	123
State	Pro = SystemProcessor; db = SystemDB;	Pro = SystemProcessor; db = SystemDB;
Expected Output	[db.baseline, db.dailyLimit, db.hourlyLimit, db.bolus, db.interval, db.password]	[0,0,0,0,0,'Invalid password!']

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

T1.2.12: Test getBaseline()

```
    function baseline = getBaseline(DB)
        % Get the injecting baseline of the injection system.
        %   Input:  None
        %   Return: baseline [float]
        %             the injecting baseline of the injection system.
        baseline = DB.baseline; % Statement - Tcover1.2.12.1
    end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.2.12.1
Coverage Item	Tcover1.2.12.1
Input	------
State	Pro = SystemProcessor; db = SystemDB; Set db.baseline = 0.03
Expected Output	0.03

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

T1.2.13: Test getDailyLimit()

```
function dailyLimit = getDailyLimit(DB)
    % Get the painkiller daily limit of the injection system.
    %   Input:  None
    %   Return: dailyLimit [float]
    %             the painkiller daily limit of the injection
    %             system.
    dailyLimit = DB.dailyLimit;   % Statement - Tcover1.2.13.1
end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.2.13.1
Coverage Item	Tcover1.2.13.1
Input	------
State	Pro = SystemProcessor; db = SystemDB; Set db. dailyLimit = 4
Expected Output	4

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

T1.2.14: Test getHourlyLimit()

```
function hourlyLimit = getHourlyLimit(DB)
    % Get the painkiller hourly limit of the injection system.
    %   Input:  None
    %   Return: hourlyLimit [float]
    %             the painkiller hourly limit of the injection
    %             system.
    hourlyLimit = DB.hourlyLimit; % Statement - Tcover1.2.14.1
end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.2.14.1
Coverage Item	Tcover1.2.14.1
Input	------
State	Pro = SystemProcessor; db = SystemDB; Set db. hourlyLimit = 4
Expected Output	4

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

T1.2.15: Test getInterval()

```
        function interval = getInterval(DB)
            % Get the compulsory bolus interval of the injection system.
            %   Input:  None
            %   Return: interval [float]
            %             the bolus interval of the injection system. unit
            %             is minute.
            interval = DB.interval; % Statement - Tcover1.2.15.1
        end
    end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.2.15.1
Coverage Item	Tcover1.2.15.1
Input	------
State	Pro = SystemProcessor; db = SystemDB; Set db. interval = 4
Expected Output	4

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

T1.2.16: Test addPainkiller()

```
        function addPainkiller(DB, time, amount)
            % Add the painkiller to amount.
            %   Input:  amount [float]
            %             set the remaining painkiller to amount (not add).
            %   Return: None
            DB.painkillerRemaining = amount; % Statement - Tcover1.2.16.1
            DB.painkillerCheckpoint = time;
        end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.2.16.1
Coverage Item	Tcover1.2.16.1
Input	datetime(),1
State	Pro = SystemProcessor; db = SystemDB;
Expected Output	db.painkillerRemaining == 1; db.painkillerCheckpoint == datetime()

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

T1.2.17: Test updateInjectionSpeed()

```
        function updateInjectionSpeed(DB, time, injectionSpeed)
            % If injectionSpeed differs from current, update the database
            % normal injection status.
            %   Input:  time [datetime]
            %             current time in processor.
            %           injectionSpeed [float]
            %             the injection speed at current time that the
            %             processor want to set.
            %   Return: None

            % First data line
            if isempty(DB.normalDataList) % Branch - Tcover1.2.17.1
                if injectionSpeed ~= 0 % Branch - Tcover1.2.17.2
                    DB.createNormalData(time, injectionSpeed);
                end
            % Already have data, currently not 0
            elseif DB.normalDataList(end).endTime == DB.DUMMYTIME &&...
                    DB.normalDataList(end).speed ~= injectionSpeed
                  % Branch - Tcover1.2.17.3
                DB.normalDataList(end).endTime = time;
                if injectionSpeed ~= 0 % Branch - Tcover1.2.17.4
                    DB.createNormalData(time, injectionSpeed);
                end
            % Already have data, currently 0
            elseif DB.normalDataList(end).endTime ~= DB.DUMMYTIME &&...
                    injectionSpeed ~= 0   % Branch - Tcover1.2.17.5
                DB.createNormalData(time, injectionSpeed);
            end
        end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.2.17.1	Test Case T1.2.17.2	Test Case T1.2.17.3
Coverage Item	Tcover1.2.17.1, Tcover1.2.17.2	Tcover1.2.17.1, Tcover1.2.17.2	Tcover1.2.17.1, Tcover1.2.17.3, Tcover1.2.17.4
Input	datetime(), 0	datetime(), 0.02	datetime(), 0.02
State	Pro=SystemProcessor; db=SystemDB;	Pro=SystemProcessor; db=SystemDB;	Pro=SystemProcessor; db=SystemDB; nd=NormalData; nd.startTime=datetime(); nd.endTime =db.DUMMYTIME; nd.speed=0.03; db.normalDataList=[nd];
Expected Output	db.normalDataList is empty	db.normalDataList(end).speed=0.02	db.normalDataList(end).speed = 0.02
	Test Case T1.2.17.4	Test Case T1.2.17.5	Test Case T1.2.17.6

Coverage Item	Tcover1.2.17.1, Tcover1.2.17.3, Tcover1.2.17.4	Tcover1.2.17.1, Tcover1.2.17.3, Tcover1.2.17.5	Tcover1.2.17.1, Tcover1.2.17.3, Tcover1.2.17.5
Input	datetime(), 0	datetime(), 0.02	datetime(), 0
State	Pro=SystemProcessor; db=SystemDB; nd=NormalData; nd.startTime=datetime(); nd.endTime =db.DUMMYTIME; nd.speed=0.03; db.normalDataList=[nd];	Pro=SystemProcessor; db=SystemDB; nd=NormalData; nd.startTime=datetime(); nd.endTime =datetime(); nd.speed=0.03; db.normalDataList=[nd];	Pro=SystemProcessor; db=SystemDB; nd=NormalData; nd.startTime=datetime(); nd.endTime =datetime(); nd.speed=0.03; db.normalDataList=[nd];
Expected Output	db.normalDataList(end).speed = 0.03	db.normalDataList(end).speed = 0.02	db.normalDataList(end).speed = 0.03

- Test coverage: 16/16=100%
- Test result: 6 passed

T1.2.18: Test makeBolus()

```
function makeBolus(DB, time)
    % Make a bolus. Write into database.
    %   Input:  time [datetime]
    %             current time in processor.
    %   Return: None

    bo = BolusData; % Statement - Tcover1.2.18.1
    bo.injectTime = time;
    bo.amount = DB.getPossibleBolus(time);
    DB.bolusDataList = [DB.bolusDataList; bo];
end
```

- Coverage Criteria: Statement coverage
- Test case

	Test Case T1.2.18.1
Coverage Item	Tcover1.2.18.1
Input	datetime()
State	Pro = SystemProcessor; db = SystemDB;
Expected Output	Length of db.bolusDataList is 1; db.bolusDataList(end).amount == db.bolus;

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

T1.2.19: Test setParameters()

```
        function setParameters(DB, baseline,dailyLimit,...
                hourlyLimit,bolus,...
                interval,password,originPassword)
            % Set the database parameters.
            %   Caller: SettingUI.ConfirmButtonPushed()
            %   Input:  baseline [float]
            %           dailyLimit [float]
            %           hourlyLimit [float]
            %           bolus [float]
            %           interval [float]
            %           password [string]
            %             the parameters to set.
            %           originPassword [string]
            %             the original password UI gives, waiting to check.
            %   Return: None

            passed = DB.checkPassword(originPassword);
            if passed   % Branch - Tcover1.2.19.1
                DB.baseline    = baseline;
                DB.dailyLimit  = dailyLimit;
                DB.hourlyLimit = hourlyLimit;
                DB.bolus       = bolus;
                DB.interval    = interval;
                DB.password    = password;
            end
        end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.2.19.1	Test Case T1.2.19.2
Coverage Item	Tcover1.2.19.1	Tcover1.2.19.1
Input	[db.baseline, db.dailyLimit, db.hourlyLimit, db.bolus, db.interval, '2333', db.password]	[db.baseline, db.dailyLimit, db.hourlyLimit, db.bolus, db.interval, '2333', '2333']
State	Pro = SystemProcessor; db = SystemDB;	Pro = SystemProcessor; db = SystemDB;
Expected Output	db.password == '2333'	db.password ~= '2333'

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

T1.3: MainBoardUI Unit Test

T1.3.1: Test show()

```
        function show(app, time, injectionSpeed, last1h, last24h,...
                      possibleBolus, interval, lastTime, painkillerLeft)
            % Update the main app's screen to show the data
            app.baseLineNum.Text = sprintf('%0.2f', injectionSpeed);
            app.Date.Text = string(datetime(time,'Format','yyyy-MM-dd'));
            app.Time.Text = string(datetime(time,'Format','HH:mm'));
            app.last1h.Text = last1h;
```

```
            app.last24h.Text = last24h;
            app.bolus.Text = sprintf('%0.2f', possibleBolus);
            if possibleBolus == 0 || strcmp(app.Switch.Value, 'Off')
            % Branch - Tcover1.3.1.1
                app.Lamp.Color = [1,0,0];
            else % Branch - Tcover1.3.1.2
                app.Lamp.Color = [0,1,0];
            end
            app.intervalTime.Text = sprintf('%0.0f min', interval);
            app.lastTime.Text = lastTime;
            app.PainKillerLeftGauge.Value = painkillerLeft;
            if painkillerLeft == 0  % Branch - Tcover1.3.1.3
                app.PainkillerWarning.Visible = 'on';
                app.PainkillerWarning.Text = "Warning: No painkiller left.";
                app.PainkillerWarning.FontColor = [1.00,0,0];
            elseif painkillerLeft < 1 % Branch - Tcover1.3.1.4
                app.PainkillerWarning.Visible = 'on';
                app.PainkillerWarning.Text = "Warning: Painkiller running
out.";
                app.PainkillerWarning.FontColor = [1.00,0.41,0.16];
            else  % Branch - Tcover1.3.1.5
                app.PainkillerWarning.Visible = 'off';
            end
            if painkillerLeft == 10 % Branch - Tcover1.3.1.6
                app.AddPainkillerButton.Enable = 'off';
            else  % Branch - Tcover1.3.1.7
                app.AddPainkillerButton.Enable = 'on';
            end
            app.SpeedLabel.Text = app.processor.getCurSpeed() + "x";
        end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.3.1.1	Test Case T1.3.1.2	Test Case T1.3.1.3
Coverage Item	Tcover1.3.1.1, Tcover1.3.1.3, Tcover1.3.1.7	Tcover1.3.1.2, Tcover1.3.1.4, Tcover1.3.1.7	Tcover1.3.1.2, Tcover1.3.1.5, Tcover1.3.1.6
Input	datetime(), 0.02, '0.80ml / 1.00ml', '1.20ml /1.00ml',1,1,string(datetime()), 0	datetime(), 0.02, '0.80ml / 1.00ml', '1.20ml /1.00ml',1,1,string(datetime()), 0.5	datetime(), 0.02, '0.80ml / 1.00ml', '1.20ml /1.00ml',1,1,string(datetime()), 10
State	Pro = SystemProcessor; db = SystemDB; mainApp = MainBoardUI;	Pro = SystemProcessor; db = SystemDB; mainApp = MainBoardUI; mainApp.Switch.Value = "On";	Pro = SystemProcessor; db = SystemDB; mainApp = MainBoardUI; mainApp.Switch.Value = "On";
Expected Output	mainApp.Lamp.Color == [1,0,0];	mainApp.Lamp.Color == [0,1,0];	mainApp.Lamp.Color == [0,1,0];

	mainApp.PainkillerWarning.FontColor==[1.00,0,0]; mainApp.AddPainkillerButton.Enable == 'on'	mainApp.PainkillerWarning.FontColor[1.00,0.41,0.16]; mainApp.AddPainkillerButton.Enable == 'on'	mainApp.PainkillerWarning.Visible=='off'; mainApp.AddPainkillerButton.Enable=='off'

- Test coverage: 9/9=100%
- Test result: 3 passed

T2: Integration Test

Since the painkiller system is so sensitive to time that there may be some deviation between the actual value and the theoretical value.

T2.1: SystemProcessor+SystemDB+mainBoardUI Integration

```
function injection_test(tc)   %T2.1
 % -------- START INJECTION -----------
     tc.choose(tc.mainApp.Switch, 'On');
     pause(15);
     tc.press(tc.bolusApp.BolusButton);
     pause(1);
     tc.press(tc.bolusApp.BolusButton);
     pause(1);
     tc.press(tc.bolusApp.BolusButton);
     pause(6);

     tc.press(tc.bolusApp.BolusButton);   pause(8);
     tc.press(tc.mainApp.AddPainkillerButton); pause(5);
     % speed up to go through the 24h
     tc.drag(tc.mainApp.TimeElapseSpeedSlider, 1.56239, 2);
     % from 600x to 3600x
     pause(18);

 end
```

- Test case

	Test Case T2.1
Coverage Item	Tcover 1.1.1.1-1.1.1.3 Tcover 1.1.2.1-1.1.2.2 Tcover 1.1.3.1-1.1.5.1 Tcover 1.1.6.1-1.1.6.2 Tcover 1.1.7.1-1.1.8.1 Tcover 1.2.1.1 Tcover 1.2.2.1-1.2.2.3 Tcover 1.2.5.1-1.2.8.1 Tcover 1.2.9.1-1.2.9.2 Tcover 1.2.12.1-1.2.16.1 Tcover 1.2.17.1-1.2.17.5 Tcover 1.2.18.1 Tcover1.3.1.1-1.3.1.7
Input	Press switch button to turn on the system.

	After 15s (150min), press bolus button. After 1s (10min), press bolus button again. After 1s (10min), press bolus button again. After 6s (60min), press bolus button again. After 8s (80min), press addPainkiller button. After 5s (50min), set the speed to 3600. Wait 18s (180min).
State	Speed is initially 600x, then set to 3600x to cross 24h. baseline = 0.01; dailyLimit = 3; hourlyLimit = 1; bolus = 0.3; interval = 15;
Expected Output	The last 1h amount will increase at beginning and stop at 0.60/1.00 for a while. After first bolus, it will increase again to 0.90/1.00. The second bolus press is invalid because it is within the interval. The third bolus will change last 1h amount to 0.99/1.00; The 4th bolus will accelerate the last 24h amount to 3.00/3.00, then the system stop injecting; After press addpainKiller button, the painkiller remaining will be set to full. After waiting enough time, the system continuous to inject.

- Test coverage: 40/40=100%
- Test result: 1 passed

T2.2: SystemProcessor+SystemDB+mainBoardUI+PasswordUI+SettingUI Integration

```
        function setting_test(tc)   %T2.2
            tc.press(tc.mainApp.SettingButton);
            % Wrong password
            tc.type(tc.mainApp.passwordDialog.InputPassword, '1234');
            tc.press(tc.mainApp.passwordDialog.ConfirmButton);
            % Correct password
            tc.type(tc.mainApp.passwordDialog.InputPassword, '123456');
            tc.press(tc.mainApp.passwordDialog.ConfirmButton);
% -------------------------- SETTING -------------------------------------
            tc.drag(tc.mainApp.settingDialog.BaseLineSlider, 0.01, 0.025);
            tc.drag(tc.mainApp.settingDialog.BolusSlider, 0.3, 0.23);
            tc.type(tc.mainApp.settingDialog.LimitperdaySpinner, 10);
            tc.type(tc.mainApp.settingDialog.LimitperhourSpinner, 2);
            tc.type(tc.mainApp.settingDialog.IntervaltimeSpinner, 20);
            tc.type(tc.mainApp.settingDialog.PasswordEditField, 'qwqqqq');
            tc.press(tc.mainApp.settingDialog.ConfirmButton);
            tc.press(tc.mainApp.settingDialog.closeButton);
            % Test changed setting
            tc.press(tc.mainApp.SettingButton);
            tc.type(tc.mainApp.passwordDialog.InputPassword, 'qwqqqq');
            tc.press(tc.mainApp.passwordDialog.ConfirmButton);
            tc.addTeardown(@delete, tc.mainApp.settingDialog);
        end
```

- Test case

	Test Case T2.2
Coverage Item	Tcover 1.1.1.2

	Tcover 1.1.1.2, 1.1.2.1, 1.1.3.1, 1.2.10.2, 1.2.11.1, 1.2.11.2, 1.2.19.1, 1.3.1.1, 1.3.1.5, 1.3.1.7
Input	Press setting button; Type wrong password; Press confirm button; Type correct password; Press confirm button; Change baseline: 0.01→0.025, bolus:0.3→0.23,dailyLimit→10, hourlyLimit→2, interval→20, password→'qwqqqq'; Press confirm button; Close setting UI; Press setting button; Type changed password; Press confirm button;
State	mainApp = MainBoardUI; Initial setting is : baseline = 0.01; dailyLimit = 3; hourlyLimit = 1; bolus = 0.3; interval = 15;
Expected Output	After press setting button, password UI will pop first. After entering wrong password, the UI will show 'Invalid password! Please try again.'. After entering correct password, the setting UI will be opened. After change the parameters, when press setting UI next time, the corrcte password will become our change password 'qwqqqq', and the parameters' value will be correctly illustrated on SettingUI.

- Test coverage: 10/10=100%
- Test result: 1 passed

T3: Functional Test

T3.1: Use Case "Get basic information"

T3.1.1: Test 'Get default information'

- Test case

	Test Case T3.1.1
State	baseline = 0.01; dailyLimit = 3; hourlyLimit = 1;
Operation	Turn on the system, pause 1 second
Expected Behavior	The MainBoardUI will show baseline/bolus/daily_limit/hourly_limit/lockout_interval/remaining_painkiller, and use lamp to show whether can inject bolus now: before turn on the system, the lamp is red, after that, the lamp becomes green.

- Test result: 1 passed

T3.1.2: Test 'Get changed information'

- Test case

	Test Case T3.1.2
State	baseline = 0.01; dailyLimit = 3; hourlyLimit = 1;
Operation	Turn on the system, pause 1 second, press bolus button, pause 2 seconds
Expected Behavior	The MainBoardUI will show baseline/bolus/daily_limit/hourly_limit/lockout_interval/remaining_painkiller, and use lamp to show whether can inject bolus now: before turn on the system, the lamp is red, after that, the lamp becomes green.

	After pressing the bolus, the lamp will become red again, and MainBoardUI will give the time of last bolus.

- Test result: 1 passed

T3.2: Use Case "Get static data"

T3.2.1: Test 'Get normal static data'

- Test case

	Test Case T3.2.1
State	baseline = 0.01; dailyLimit = 3; hourlyLimit = 1;
Operation	Turn on the system, open MoniterUI, pause 1 second
Expected Behavior	The MainBoardUI will show amount of painkiller injected in last hour and last day, MonitorUI can show injection history with figure and tables.

- Test result: 1 passed

T3.2.2: Test 'Get special static data'

- Test case

	Test Case T3.2.2
State	baseline = 0.01; dailyLimit = 3; hourlyLimit = 1;
Operation	Turn on the system, open MoniterUI, pause 1 second, press bolus button, pause 3 second, set the speed to 3600x, pause 3 second, set the accuracy number of sampled to 600x on MonitorUI.
Expected Behavior	The MainBoardUI will show amount of painkiller injected in last hour and last day, MonitorUI can show injection history with figure and tables. After making a bolus, there will be a spike on the dynamic figure on MoniterUI. After set the speed to 3600x, the axis of the figure will be compressed. After set the axis accuracy to 60x in MonitorUI, the figure will be stressed.

- Test result: 1 passed

T3.3: Use Case "Set parameters"

T3.3.1: Test 'Set parameters'

- Test case

	Test Case T3.3.1
State	mainApp = MainBoardUI; Initial setting is : baseline = 0.01; dailyLimit = 3; hourlyLimit = 1; bolus = 0.3; interval = 15;
Operation	Press setting button; Wait until the amount injected in last 1h reached a positive number. Type wrong password; Press confirm button; Type correct password; Press confirm button;

	Change baseline: 0.01→0.025, bolus:0.3→0.23,dailyLimit→10, hourlyLimit→0, interval→20, password→'qwqqqq'; Press confirm button; Close setting UI; Press setting button; Type changed password; Press confirm button;
Expected Behavior	After press setting button, password UI will pop first. After entering wrong password, the UI will show 'Invalid password! Please try again.'. After entering correct password, the setting UI will be opened. After change the parameters, we will see the amount injected in last 1h and last 24h is also updated to 0 to ensure they are within limit; When press setting UI next time, the corrcte password will become our change password 'qwqqqq', and the parameters' value will be correctly illustrated on SettingUI.

- Test result: 1 passed

T3.4: Use Case "Inject bolus"

T3.4.1: Test 'Inject bolus'

- Test case

	Test Case T3.4.1
State	baseline = 0.01; dailyLimit = 3; hourlyLimit = 1;
Operation	Turn on the system, pause 1s, press bolus button, pause 1s, pressbolus button again.
Expected Behavior	The first bolus is valid, the 2nd bolus is invalis because it is too close to 1st one.

- Test result: 1 passed

T4: Model Checking

A UPPAAL model of this PainKiller system is built for model checking.

Full PainKiller Model

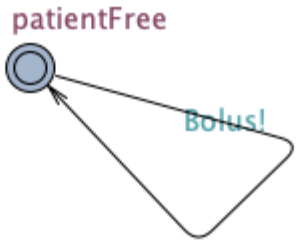
As UPPAAL is weak to check 'double' type, so I multiplied all the numbers involved in the dose by 100 times: baseline 0.01→10; dailyLimit 3→300; hourlyLimit 1→100; bolus 0.03→30;

A 24*60 injectHistory array is used for recording the injection history of every minute over the past 24*60 minutes (1 day).

Physician

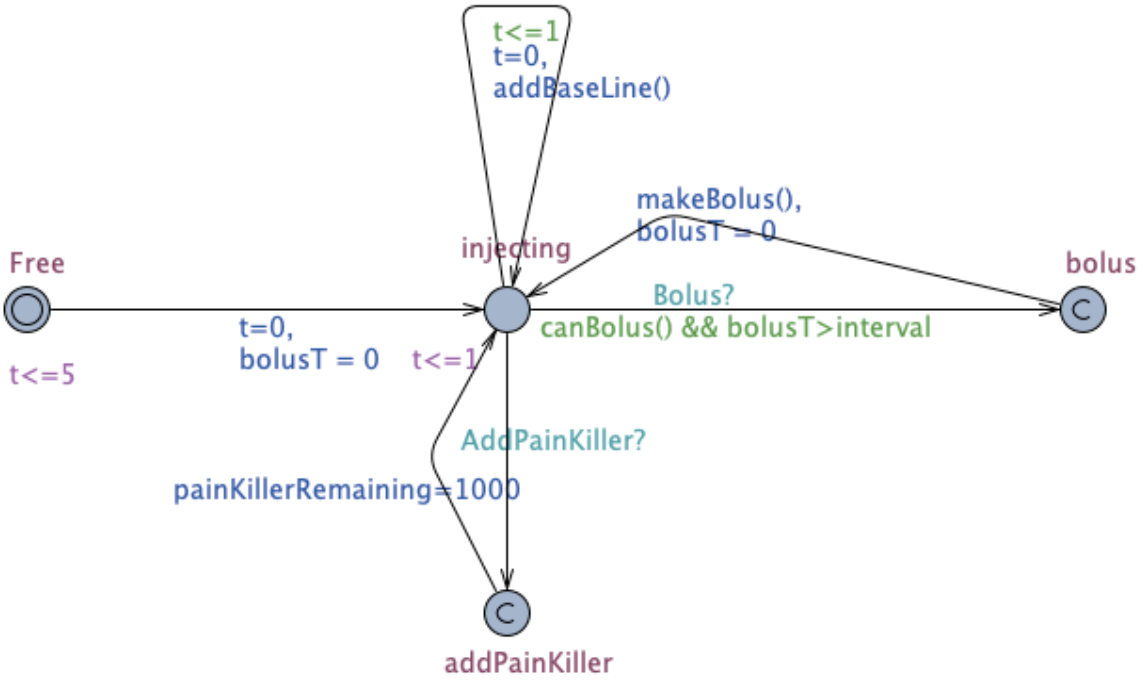
Physician will help to add painkiller so that our model checking can continue. As soon as the remaining painkiller == 200, physician will call ‘AddPainKiller!’ and painkiller will be added to 1000.

Patient



Just the same as real life, the patient can call bolus whenever he wants.

Processor



The processor will not stay at initial state more than 5s to avoid deadlock.

The injecting state will been updated every 1 second (which represents 1min in real life). The function 'addBaseLine()' will calculate the baseline can be added now and update the injectHistory array.

When processor receives 'Bolus' call, it will check if a bolus is valid under current state. If it is valid, bolus will been made and the injectHistory array will be updated at the same time. A clock 'bolusT' is specifically used to record the interval between two success bolus.

When processor receives 'AddPainKiller' call, it will immediately set the painkillerRemaing to full.

Check Properties

P4.1

Property	A[] processor.last1h() <= hourlyLimit
Description	At any time, the injection volume in the past 1 hour will not exceed the limit.
Result	Passed

P4.2

Property	A[] processor.last24h() <= dailyLimit
Description	At any time, the injection volume in the past 24 hour will not exceed the limit.
Result	Passed

P4.3

Property	A[] processor.bolus imply processor.bolusT >= interval
Description	The time between two successful bolus will always be greater than the setting interval.
Result	Passed

P4.4

Property	A[] 0 <= painKillerRemaining <= 1000
Description	The painkiller margin must always be within the limits.
Result	Passed