SOFTWARE VALIDATIONS

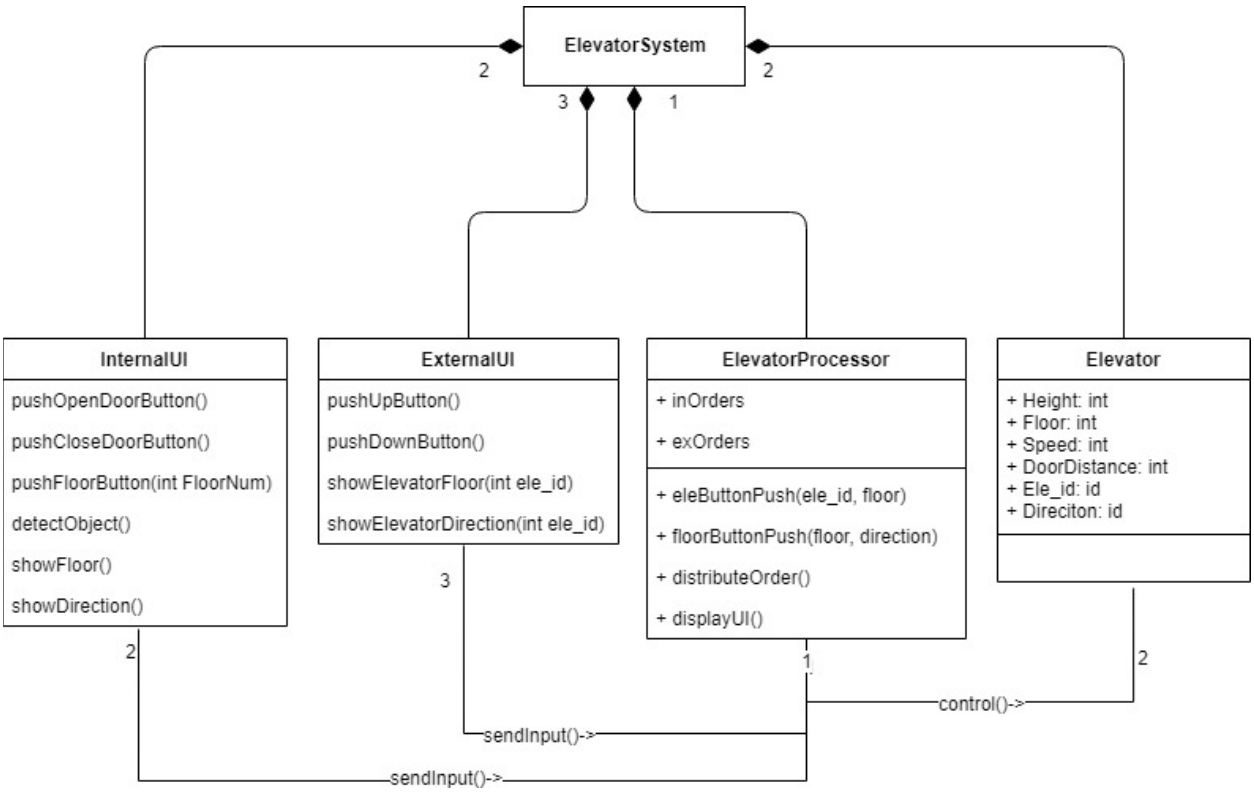
Elevator System

Group 16
Author: Haoyi Wu

Table of Contents

System Architecture

The system architecture is shown below:



T1: Unit Test

T1.1: elevatorProcessor Unit Test

T1.1.1: Test eles()

```
function result=eles(pro, id)
    % Method to get the elevator of id
    if(id==1)                              % Branch – Tcover1.1.1.1
        result = pro.ele1;
    elseif(id==2)                          % Branch – Tcover1.1.1.2
        result = pro.ele2;
    end
end
```

- Coverage Criteria: Branch coverage
  - Note: Tcover1.1.1.1 has 2 sub coverage items: Tcover1.1.1.1.1 for branch taken true, Tcover1.1.1.1.2 for branch taken false. All coverage items follow the same naming rule.
- Test case

	Test Case T1.1.1.1	Test Case T1.1.1.2	Test Case T1.1.1.3
Coverage Item	Tcover1.1.1.1.1	Tcover1.1.1.1.2 Tcover1.1.1.2.1	Tcover1.1.1.2.2

Input	1	2	3

State	pro=elevatorProcessor;
	pro.ele1 = 1;
	pro.ele2 = 2;
		pro=elevatorProcessor;
		pro.ele1 = 1;
		pro.ele2 = 2;
			pro=elevatorProcessor;
			pro.ele1 = 1;
			pro.ele2 = 2;

Expected Output	ele==pro.ele1	ele==pro.ele2	'MATLAB:unassignedOutputs'

- Test coverage: 4/4=100%
- Test result: 3 passed

T1.1.2: Test checkFloor()

```
function checkFloor(pro, id)
    %Method to check the current floor of id_th elevator, which is
    %called in the stateflow.
   if pro.eles(id).height<=2.5          % Branch - Tcover1.1.2.1
       pro.eles(id).floor=1;
   elseif pro.eles(id).height>7.5       % Branch - Tcover1.1.2.2
       pro.eles(id).floor=3;
   else
       pro.eles(id).floor=2;
   end
   pro.displayUI();
end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.1.2.1	Test Case T1.1.2.2	Test Case T1.1.2.3
Coverage Item	Tcover1.1.2.1.1	Tcover1.1.2.1.2 Tcover1.1.2.2.1	Tcover1.1.2.2.2
Input	1	2	1
State	pro=elevatorProcessor; pro.ele1 = elevator_1('processor', pro, 'floor', 3, 'id', 1, 'height', 1.1);	pro=elevatorProcessor; pro.ele2 = elevator_2('processor', pro, 'floor', 1, 'id', 2, 'height', 8.0);	pro=elevatorProcessor; pro.ele1 = elevator_1('processor', pro, 'floor', 3, 'id', 1, 'height', 4.6);
Expected Output	pro.ele1.floor==1	pro.ele2.floor==3	pro.ele1.floor==2

- Test coverage: 4/4=100%
- Test result: 3 passed

T1.1.3: Test checkSpeed()

```
function checkSpeed(pro, id)
    %Method to check the current speed of id_th elevator and
    %transfer its state if needed, which is called in the
    %stateflow.
   if pro.eles(id).v*pro.eles(id).direction <= 0    % Branch -
Tcover1.1.3.1
       STOP(pro.eles(id))
   elseif abs(pro.eles(id).v) >= pro.maxSpeed       % Branch -
Tcover1.1.3.2
       MAXV(pro.eles(id))
```

```
        end
    end
```

- Coverage Criteria: Branch coverage
- Test case
    - Note: For testing, dummy elevator is used. In real elevator model, parameters like stop and maxv will not occur. Same applies to the following testing.

	Test Case T1.1.3.1	Test Case T1.1.3.2	Test Case T1.1.3.3
Coverage Item	Tcover1.1.3.1.1	Tcover1.1.3.1.2 Tcover1.1.3.2.1	Tcover1.1.3.2.2
Input	1	2	1

State	pro=elevatorProcessor;
	pro.ele1 = DummyElevator('processor', pro, 'floor', 1, 'id', 1, 'height', 1.0);
	pro.ele1.v = 0;
	pro.ele1.direction = -1;
		pro=elevatorProcessor;
		pro.ele2 = DummyElevator('processor', pro, 'floor', 1, 'id', 1, 'height', 1.0);
		pro.ele2.v = 1;
		pro.ele2.direction = 1;
			pro=elevatorProcessor;
			pro.ele1 = DummyElevator('processor', pro, 'floor', 3, 'id', 1, 'height', 8.0);
			pro.ele1.v = -0.5;
			pro.ele1.direction = -1;

Expected Output	pro.ele1.stop==1	pro.ele2.maxv==1	pro.ele1.stop==0 pro.ele1.maxv==0

- Test coverage: 4/4=100%
- Test result: 3 passed

T1.1.4 Test checkDes()

```
function checkDes(pro, id)
    %Method to check whether the elevator need to slow down, which
    %is called in the stateflow.

    orders =
pro.inOrders(id,:)+pro.exOrders_dis(1,:,id)+pro.exOrders_dis(2,:,id);
    for i=1:3
        if orders(i)                    % Branch - Tcover1.1.4.1
            if (pro.eles(id).height-pro.height*(i-
1))*pro.eles(id).direction>=-1*pro.limitHeight % Branch - Tcover1.1.4.2
                SLOW(pro.eles(id));
                break
            end
        end
    end
end
```

- Coverage Criteria: Branch coverage
    - Branch coverage should include for loop, but that increase the workload a lot. Since most for loop will cover both conditions, the test cases will by default cover for loop branch, but not stated in coverage item.
- Test case

	Test Case T1.1.4.1
Coverage Item	Tcover1.1.4.1.1, Tcover1.1.4.1.2, Tcover1.1.4.2.1, Tcover1.1.4.2.2
Input	1

State	pro=elevatorProcessor;
	pro.height = 5;
	pro.limitHeight = (pro.maxSpeed/(0.25/10) - 1) * pro.maxSpeed / 20;
	pro.inOrders = [0 1 0; 0 0 0];
	
	pro.ele1 = DummyElevator('processor', pro, 'floor', 1, 'id', 1, 'height', 4.0);
	pro.ele1.v = 1;
	pro.ele1.direction = 1;

Expected Output	pro.ele1.slow==1

- Test coverage: 4/4=100%
- Test result: 1 passed

T1.1.5 Test checkDoorOpen()

```
function checkDoorOpen(pro, id)
    %Method to check whether the elevator need to open door when stopped,
which
    %is called in the stateflow.
    pro.distributeOrder()
    if pro.eles(id).direction ~= 0  % Branch - Tcover1.1.5.1
        dir = 1.5 + pro.eles(id).direction/2;
        orders = pro.inOrders(id,:)+pro.exOrders(dir,:);
        if orders(pro.eles(id).floor)   % Branch - Tcover1.1.5.2
            pro.inOrders(id, pro.eles(id).floor)=0;
            pro.exOrders(dir, pro.eles(id).floor)=0;
            pro.exOrders_dis(dir, pro.eles(id).floor,1)=0;
            pro.exOrders_dis(dir, pro.eles(id).floor,2)=0;
            pro.eles(id).openTime=0;
            OPEN(pro.eles(id))
        elseif pro.exOrders_dis(1,pro.eles(id).floor,id)
            % Branch - Tcover1.1.5.3
            pro.exOrders_dis(1, pro.eles(id).floor,id)=0;
            pro.exOrders(1, pro.eles(id).floor)=0;
            pro.eles(id).openTime=0;
            pro.eles(id).direction= -1;
            OPEN(pro.eles(id))
        elseif pro.exOrders_dis(2,pro.eles(id).floor,id)
            % Branch - Tcover1.1.5.4
            pro.exOrders_dis(2, pro.eles(id).floor,id)=0;
            pro.exOrders(2, pro.eles(id).floor)=0;
            pro.eles(id).openTime=0;
            pro.eles(id).direction= 1;
            OPEN(pro.eles(id))
        end
    else
        if pro.exOrders_dis(1,pro.eles(id).floor,id)
            % Branch - Tcover1.1.5.5
            dir = 1;
            pro.eles(id).direction = -1;
            pro.inOrders(id, pro.eles(id).floor)=0;
            pro.exOrders(dir, pro.eles(id).floor)=0;
            pro.exOrders_dis(dir, pro.eles(id).floor, 1)=0;
```

```
            pro.exOrders_dis(dir, pro.eles(id).floor, 2)=0;
            pro.eles(id).openTime=0;
            OPEN(pro.eles(id))
        elseif pro.exOrders_dis(2,pro.eles(id).floor,id)
      % Branch - Tcover1.1.5.6
            dir = 2;
            pro.eles(id).direction = 1;
            pro.inOrders(id, pro.eles(id).floor)=0;
            pro.exOrders(dir, pro.eles(id).floor)=0;
            pro.exOrders_dis(dir, pro.eles(id).floor, 1)=0;
            pro.exOrders_dis(dir, pro.eles(id).floor, 2)=0;
            pro.eles(id).openTime=0;
            OPEN(pro.eles(id))
        elseif pro.inOrders(id,pro.eles(id).floor)
      % Branch - Tcover1.1.5.7
            if pro.eles(id).height == pro.height*(pro.eles(id).floor-1)
            % Branch - Tcover1.1.5.8
                pro.inOrders(id, pro.eles(id).floor)=0;
                pro.eles(id).openTime=0;
                OPEN(pro.eles(id))
            end
        end
    end
    if pro.eleUIs(id).openDoor.BackgroundColor(3)<=0.1
    % Branch - Tcover1.1.5.9
        if pro.eles(id).doorDis<1
            % Branch - Tcover1.1.5.10
            OPEN(pro.eles(id))
        end
    elseif pro.eleUIs(id).detectObj.BackgroundColor(2)<=0.1 &&...
               pro.eleUIs(id).detectObj.BackgroundColor(3)<=0.1
            % Branch - Tcover1.1.5.11
        if pro.eles(id).doorDis<1
      % Branch - Tcover1.1.5.12
            OPEN(pro.eles(id))
        end
    end
    pro.displayUI();
end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.1.5.1	Test Case T1.1.5.2	Test Case T1.1.5.3	Test Case T1.1.5.4
Coverage Item	Tcover1.1.5.1.1, Tcover1.1.5.2.1, Tcover1.1.5.9.2, Tcover1.1.5.11.2	Tcover1.1.5.2.2, Tcover1.1.5.3.1	Tcover1.1.5.3.2, Tcover1.1.5.4.1	Tcover1.1.5.4.2, Tcover1.1.5.9.1, Tcover1.1.5.10.2
Input	1	1	1	1
State	elevator 1 on v=0 and direction upwards, reaches	elevator 1 on v=0 and direction upwards, reaches floor 2, with order	elevator 1 on v=0 and direction upwards, reaches floor 2, with order	elevator 1 on v=0 and direction upwards, reaches floor 2, with no

	floor 2, with interal order on floor 2	distributed on floor 2 down	distributed on floor 2 up	order. Open door button pressed and door is open.
Expected Output	pro.ele1.open==1	pro.ele1.open==1	pro.ele1.open==1	pro.ele1.open==0
	Test Case T1.1.5.5	Test Case T1.1.5.6	Test Case T1.1.5.7	Test Case T1.1.5.8
Coverage Item	Tcover1.1.5.1.2, Tcover1.1.5.5.1	Tcover1.1.5.5.2, Tcover1.1.5.6.1	Tcover1.1.5.6.2, Tcover1.1.5.7.1, Tcover1.1.5.8.1	Tcover1.1.5.8.2
Input	2	2	2	2
State	elevator 2 on v=0 and stopped, reaches floor 3, with order distributed on floor 3 down.	elevator 2 on v=0 and stopped, reaches floor 2, with order distributed on floor 2 up.	elevator 2 on v=0 and stopped, reaches floor 2, with internal order on floor 2. Height aligned correctly.	elevator 2 on v=0 and stopped, reaches floor 2, with internal order on floor 2. Height aligned incorrectly.
Expected Output	pro.ele2.open==1	pro.ele2.open==1	pro.ele2.open==1	pro.ele2.open==0
	Test Case T1.1.5.9	Test Case T1.1.5.10	Test Case T1.1.5.11	
Coverage Item	Tcover1.1.5.10.1	Tcover1.1.5.11.1, Tcover1.1.5.12.1	Tcover1.1.5.12.2	
Input	2	2	2	
State	elevator 2 on v=0 and stopped, reaches floor 2, with no order. Open door button pressed and door is closed.	elevator 2 on v=0 and stopped, reaches floor 2, with no order. Object detected and door is closed.	elevator 2 on v=0 and stopped, reaches floor 2, with no order. Object detected and door is open.	
Expected Output	pro.ele2.open==1	pro.ele2.open==1	pro.ele2.open==0	

- Test coverage: 24/24=100%
- Test result: 11 passed

T1.1.6 Test checkDoorClose()

```
%Method to check whether the elevator need to close door when stopped, which
%is called in the stateflow.
if (pro.eles(id).openTime>=5 ||
pro.eleUIs(id).closeDoor.BackgroundColor(3)<=0.9)...
        && (pro.eleUIs(id).openDoor.BackgroundColor(3)>0.9...
        && pro.eleUIs(id).detectObj.BackgroundColor(2)>0.9 )
% Branch - Tcover1.1.6.1
    CLOSE(pro.eles(id))
end
```

- Coverage Criteria: Branch coverage

- Test case

	Test Case T1.1.6.1	Test Case T1.1.6.2
Coverage Item	Tcover1.1.6.1.2	Tcover1.1.6.1.1
Input	1	2

State	pro=elevatorProcessor;
	pro.ele1 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0);
	pro.ele1.v = 0;
	
	eleUI=internalUI();
	eleUI.openDoor.BackgroundColor = [1 1 0];
	pro.eleUIs = [eleUI eleUI];
		pro=elevatorProcessor;
		pro.ele2 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0);
		pro.ele2.v = 0;
		pro.ele2.openTime = 5;
		
		eleUI=internalUI();
		pro.eleUIs = [eleUI eleUI];

Expected Output	pro.ele1.close==0	pro.ele2.close==1

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

T1.1.7 Test checkDoorDis()

```
function checkDoorDis(pro, id)
    %Method to check the current door status of id_th elevator and
    %transfer its state if needed, which is called in the
    %stateflow.
    pro.displayUI();
    if pro.eles(id).doorDis<=0          % Branch - Tcover1.1.7.1
        CLOSED(pro.eles(id))
    elseif pro.eles(id).doorDis>=1      % Branch - Tcover1.1.7.2
        OPENED(pro.eles(id))
    end
end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.1.7.1	Test Case T1.1.7.2	Test Case T1.1.7.3
Coverage Item	Tcover1.1.7.1.1	Tcover1.1.7.1.2 Tcover1.1.7.2.1	Tcover1.1.7.2.2
Input	1	1	2

State	pro=elevatorProcessor;
	pro.ele1 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0);
	pro.ele2 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0);
	pro.ele1.doorDis = 0;
		pro=elevatorProcessor;
		pro.ele1 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0);
		pro.ele2 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0);
		pro.ele1.doorDis = 1;
			pro=elevatorProcessor;
			pro.ele1 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0);
			pro.ele2 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0);
			pro.ele2.doorDis = 0.5;

Expected Output	pro.ele1.closed==1	pro.ele1.opened==1	pro.ele2.closed==0 pro.ele2.opened==0

- Test coverage: 4/4=100%
- Test result: 3 passed

T1.1.8 Test checkMove()

```
function checkMove(pro, id)
    %Method to check the whether the id_th elevator should start
    %to move, which is called in the stateflow.
    if pro.eles(id).direction==1        % Branch - Tcover1.1.8.1
        orders=pro.inOrders(id,:)+pro.exOrders_dis(2,:,id);
        for i=3:-1:1
            if orders(i)                % Branch - Tcover1.1.8.2
                if i-pro.eles(id).floor>0   % Branch - Tcover1.1.8.3
                    UP(pro.eles(id));
                    return;
                elseif i-pro.eles(id).floor<0   % Branch - Tcover1.1.8.4
                    DOWN(pro.eles(id));
                    return;
                end
            end
        end
    elseif pro.eles(id).direction==-1   % Branch - Tcover1.1.8.5
        orders=pro.inOrders(id,:)+pro.exOrders_dis(1,:,id);
        for i=1:1:3
            if orders(i)                % Branch - Tcover1.1.8.6
                if i-pro.eles(id).floor<0   % Branch - Tcover1.1.8.7
                    DOWN(pro.eles(id));
                    return;
                elseif i-pro.eles(id).floor>0   % Branch - Tcover1.1.8.8
                    UP(pro.eles(id));
                    return;
                end
            end
        end
    end


orders=pro.inOrders(id,:)+pro.exOrders_dis(1,:,id)+pro.exOrders_dis(2,:,id);
    if sum(orders)==0                   % Branch - Tcover1.1.8.9
        pro.eles(id).direction=0;
    else
        for i=1:1:3
            if orders(i)                % Branch - Tcover1.1.8.10
                if i-pro.eles(id).floor<0   % Branch - Tcover1.1.8.11
                    DOWN(pro.eles(id));
                    return;
                elseif i-pro.eles(id).floor>0   % Branch - Tcover1.1.8.12
                    UP(pro.eles(id));
                    return;
                end
            end
        end
    end
```

```
    pro.displayUI()
end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.1.8.1	Test Case T1.1.8.2	Test Case T1.1.8.3	Test Case T1.1.8.4
Coverage Item	Tcover1.1.8.1.1, Tcover1.1.8.2.2, Tcover1.1.8.2.1, Tcover1.1.8.3.1	Tcover1.1.8.3.2, Tcover1.1.8.4.1	Tcover1.1.8.4.2, Tcover1.1.8.9.2, Tcover1.1.8.10.1, Tcover1.1.8.10.2, Tcover1.1.8.11.2, Tcover1.1.8.12.2	Tcover1.1.8.1.2, Tcover1.1.8.5.1, Tcover1.1.8.6.1, Tcover1.1.8.6.2, Tcover1.1.8.7.1
Input	1	1	1	2
State	pro=elevatorProcessor; pro.ele1 = DummyElevator('processor', pro, 'floor', 1, 'id', 1, 'height', 0.0); pro.ele1.direction = 1; pro.inOrders(1,2) = 1;	pro=elevatorProcessor; pro.ele1 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0); pro.ele1.direction = 1; pro.inOrders(1,1) = 1;	pro=elevatorProcessor; pro.ele1 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0); pro.ele1.direction = 1; pro.inOrders(1,2) = 1;	pro=elevatorProcessor; pro.ele2 = DummyElevator('processor', pro, 'floor', 3, 'id', 1, 'height', 10.0); pro.ele2.direction = -1; pro.inOrders(2,2) = 1;
Expected Output	pro.ele1.up==1	pro.ele1.down==1	pro.ele1.up==0 pro.ele1.down==0	pro.ele2.down==0
	Test Case T1.1.8.5	Test Case T1.1.8.6	Test Case T1.1.8.7	Test Case T1.1.8.8
Coverage Item	Tcover1.1.8.7.2, Tcover1.1.8.8.1	Tcover1.1.8.8.2, Tcover 1.1.8.11.1	Tcover 1.1.8.12.1	Tcover 1.1.8.9.1
Input	2	2	2	2
State	pro=elevatorProcessor; pro.ele2 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0); pro.ele2.direction = -1; pro.inOrders(2,3) = 1;	pro=elevatorProcessor; pro.ele2 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0); pro.ele2.direction = -1; pro.inOrders(2,2) = 1; pro.exOrders_dis(2,1,2) = 1;	pro=elevatorProcessor; pro.ele2 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0); pro.ele2.direction = 0; pro.inOrders(2,2) = 1; pro.exOrders_dis(1,3,2) = 1;	pro=elevatorProcessor; pro.ele1 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0); pro.ele2 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0); pro.ele2.direction = 1;

Expected Output	pro.ele2.up==1	pro.ele2.down==1	pro.ele2.up==1	pro.ele2.up==0 pro.ele2.down==0 pro.ele2.direction==0

- Test coverage: 24/24=100%
- Test result: 8 passed

T1.1.9 Test eleButtonPush()

```
function eleButtonPush(pro, id, floor)
    %Called when the up/down buttons of any floor is pushed
    if pro.inOrders(id, floor)==1       % Branch - Tcover1.1.9.1
        if pro.eles(id).doorDis>0       % Branch - Tcover1.1.9.2
            pro.inOrders(id, floor)=0;
        elseif pro.eles(id).floor~=floor && sum(pro.inOrders(id, :))~=1 %
Branch - Tcover1.1.9.3
            pro.inOrders(id, floor)=0;
        end
    else
        direction=sign(floor - pro.eles(id).floor);
        if pro.eles(id).direction==0    % Branch - Tcover1.1.9.4
            pro.inOrders(id, floor) = 1;
        elseif pro.eles(id).direction==1 && direction==1    % Branch -
Tcover1.1.9.5
            pro.inOrders(id, floor) = 1;
        elseif pro.eles(id).direction==-1 && direction==-1  % Branch -
Tcover1.1.9.6
            pro.inOrders(id, floor) = 1;
        end
    end
    pro.displayUI();
end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.1.9.1	Test Case T1.1.9.2	Test Case T1.1.9.3	Test Case T1.1.9.4
Coverage Item	Tcover1.1.9.1.1, Tcover1.1.9.2.1	Tcover1.1.9.2.2, Tcover1.1.9.3.1	Tcover1.1.9.3.2	Tcover1.1.9.1.2, Tcover1.1.9.4.1
Input	1, 2	1, 1	1, 2	1, 3

State	pro=elevatorProcessor;
	pro.ele1 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0);
	pro.ele1.doorDis = 1;
		pro=elevatorProcessor;
		pro.ele1 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0);
		pro.ele1.doorDis = 0;
			pro=elevatorProcessor;
			pro.ele1 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0);
			pro.ele1.doorDis = 0;
				pro=elevatorProcessor;
				pro.ele1 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0);
				pro.ele1.doorDis = 1;

	pro.inOrders(1,2) = 1;	pro.inOrders(1,:) = [1 1 1];	pro.inOrders(1,2) = 1;	pro.ele1.direction = 0; pro.inOrders(1,2) = 1;
Expected Output	pro.inOrders(1,2)==0	pro.inOrders(1,1)==0	pro.inOrders==[0 1 0; 0 0 0]	pro.inOrders(1,3)==1
	Test Case T1.1.9.5	Test Case T1.1.9.6	Test Case T1.1.9.7	
Coverage Item	Tcover1.1.9.4.2, Tcover1.1.9.5.1	Tcover1.1.9.5.2, Tcover1.1.9.6.1	Tcover1.1.9.6.2	
Input	1, 3	1, 1	1, 1	

State	pro=elevatorProcessor;
	pro.ele1 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0);
	pro.ele1.direction = 1;
		pro=elevatorProcessor;
		pro.ele1 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0);
		pro.ele1.direction = -1;
			pro=elevatorProcessor;
			pro.ele1 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0);
			pro.ele1.direction = 1;	

Expected Output	pro.inOrders(1,3)==1	pro.inOrders(1,1)==1	pro.inOrders==[0 0 0; 0 0 0]	

- Test coverage: 12/12=100%
- Test result: 7 passed

T1.1.10 Test floorButtonPush()

```
function floorButtonPush(pro, floor, dir)
      %Called when the floor buttons in elevator is pushed
      if (pro.exOrders(dir, floor)~=1) && (floor~=1 || dir~=1) ...
              && (floor~=3 || dir~=2)     % Branch - Tcover1.1.10.1
          pro.exOrders(dir, floor)=1;
          pro.displayUI();
          pro.distributeOrder();
      end
  end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.1.10.1	Test Case T1.1.10.2
Coverage Item	Tcover1.1.10.1.1	Tcover1.1.10.1.2
Input	2, 1	2, 2
State	pro=elevatorProcessor;	pro=elevatorProcessor;

		pro.exOrders(2,2) = 1;
Expected Output	Update UI pro.exOrders(1,2)==1	pro.exOrders(2,2)==1

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

T1.1.11 distributeOrder()

```
function distributeOrder(pro)
    %Method to distribute orders to elevators
    orders = pro.exOrders-pro.exOrders_dis(:,:,1)-pro.exOrders_dis(:,:,2);
    for dir=1:2
        for floor=1:3
            if orders(dir, floor)==1        % Branch - Tcover1.1.11.1
                if abs((floor-1)*pro.height-pro.eles(1).height) > abs((floor-
1)*pro.height-pro.eles(2).height) % Branch - Tcover1.1.11.2
                    idOrder=[2, 1];
                else
                    idOrder=[1, 2];
                end
                for i=1:2
                    id=idOrder(i);
                    if pro.eles(id).direction==0    % Branch - Tcover1.1.11.3
                        pro.exOrders_dis(dir, floor, id)=1;
                        break;
                    else
                        direction = pro.eles(id).direction/2+1.5;
                        if direction==dir
                        % Branch - Tcover1.1.11.4
                            if pro.eles(id).doorDis>0
                              % Branch - Tcover1.1.11.5
                                if (floor-
pro.eles(id).floor)*pro.eles(id).direction >= 0   % Branch - Tcover1.1.11.6
                                    pro.exOrders_dis(dir, floor, id)=1;
                                    pro.exOrders_dis(3-dir, floor, id)=0;
                                    break;
                                end
                            else
                                if (floor-
pro.eles(id).floor)*pro.eles(id).direction > 0    % Branch - Tcover1.1.11.7
                                    pro.exOrders_dis(dir, floor, id)=1;
                                    pro.exOrders_dis(3-dir, floor, id)=0;
                                    break;
                                end
                            end
                        end
                    end
                end
            end
        end
    end
    pro.displayUI();
end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.1.11.1	Test Case T1.1.11.2	Test Case T1.1.11.3
Coverage Item	Tcover1.1.11.1.1, Tcover1.1.11.1.2, Tcover1.1.11.2.1, Tcover1.1.11.2.2, Tcover1.1.11.3.1, Tcover1.1.11.3.2, Tcover1.1.11.4.1, Tcover1.1.11.4.2, Tcover1.1.11.5.2, Tcover1.1.11.7.2	Tcover1.1.11.5.1, Tcover1.1.11.6.1, Tcover1.1.11.6.2	Tcover1.1.11.7.1
Input	--	--	--

State	pro=elevatorProcessor;
	pro.height = 5;
	pro.ele1 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0);
	pro.ele1.direction = 1;
	pro.ele2 = DummyElevator('processor', pro, 'floor', 3, 'id', 1, 'height', 10.0);
	pro.ele2.direction = 0;
	pro.exOrders = [0 1 1; 0 1 0];
		pro=elevatorProcessor;
		pro.height = 5;
		pro.ele1 = DummyElevator('processor', pro, 'floor', 2, 'id', 1, 'height', 5.0);
		pro.ele1.direction = 1;
		pro.ele1.doorDis = 1;
		pro.ele2 = DummyElevator('processor', pro, 'floor', 3, 'id', 1, 'height', 10.0);
		pro.ele2.direction = -1;
		pro.exOrders = [0 0 0; 1 1 0];
			pro=elevatorProcessor;
			pro.height = 5;
			pro.ele1 = DummyElevator('processor', pro, 'floor', 3, 'id', 1, 'height', 10.0);
			pro.ele1.direction = -1;
			pro.ele2 = DummyElevator('processor', pro, 'floor', 3, 'id', 1, 'height', 10.0);
			pro.ele2.direction = -1;
			pro.exOrders = [0 1 0; 0 0 0];

Expected Output	pro.exOrders_dis(1,3,2)==1	pro.exOrders_dis(2,2,1)==1	pro.exOrders_dis(1,2,1)==1

- Test coverage: 14/14=100%
- Test result: 3 passed

T1.1.12 Test displayUI()

```
function displayUI(pro)
    %Method to refresh all UIs
    for i=1:length(pro.floorUIs)
        pro.floorUIs(i).currentFloor_1.Text = num2str(pro.eles(1).floor);
        pro.floorUIs(i).currentFloor_2.Text = num2str(pro.eles(2).floor);
        pro.floorUIs(i).upLamp_1.Color = [0,0,0];
        pro.floorUIs(i).downLamp_1.Color = [0,0,0];
        pro.floorUIs(i).upLamp_2.Color = [0,0,0];
        pro.floorUIs(i).downLamp_2.Color = [0,0,0];
        if pro.eles(1).direction == 1   % Branch - Tcover1.1.12.1
            pro.floorUIs(i).upLamp_1.Color = [0,1,0];
        elseif pro.eles(1).direction == -1  % Branch - Tcover1.1.12.2
```

```
            pro.floorUIs(i).downLamp_1.Color = [0,1,0];
        end
        if pro.eles(2).direction == 1   % Branch - Tcover1.1.12.3
            pro.floorUIs(i).upLamp_2.Color = [0,1,0];
        elseif pro.eles(2).direction == -1  % Branch - Tcover1.1.12.4
            pro.floorUIs(i).downLamp_2.Color = [0,1,0];
        end

        pro.floorUIs(i).down.BackgroundColor = [1,1,1]-
[0,0,pro.exOrders(1,i)];
        pro.floorUIs(i).up.BackgroundColor = [1,1,1]-[0,0,pro.exOrders(2,i)];
    end

    for i=1:2
        pro.eleUIs(i).currentFloor.Text = num2str(pro.eles(i).floor);
        pro.eleUIs(i).upLamp.Color = [0,0,0];
        pro.eleUIs(i).downLamp.Color = [0,0,0];
        if pro.eles(i).direction == 1   % Branch - Tcover1.1.12.5
            pro.eleUIs(i).upLamp.Color = [0,1,0];
        elseif pro.eles(i).direction == -1 % Branch - Tcover1.1.12.6
            pro.eleUIs(i).downLamp.Color = [0,1,0];
        end
        pro.eleUIs(i).doorDistance.Value=pro.eles(i).doorDis;
        pro.eleUIs(i).height.Value=pro.eles(i).height;

        for j=1:3
            pro.eleUIs(i).floorLamps(j).BackgroundColor = [1,1,1]-
[0,0,pro.inOrders(i,j)];
        end
    end
    pro.displayTime();
end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.1.12.1	Test Case T1.1.12.2	Test Case T1.1.12.3
Coverage Item	Tcover1.1.12.1.1, Tcover1.1.12.3.1, Tcover1.1.12.5.1	Tcover1.1.12.1.2, Tcover1.1.12.2.1, Tcover1.1.12.3.2, Tcover1.1.12.4.1, Tcover1.1.12.5.2, Tcover1.1.12.6.1	Tcover1.1.12.2.2, Tcover1.1.12.4.2, Tcover1.1.12.6.2
Input	--	--	--

State	pro=elevatorProcessor;
	pro.ele1 =
	DummyElevator('processor',
	pro, 'floor', 2, 'id', 1, 'height',
	5.0);
	pro.ele1.direction = 1;
	pro.ele2 =
	DummyElevator('processor',
		pro=elevatorProcessor;
		pro.ele1 =
		DummyElevator('processor',
		pro, 'floor', 2, 'id', 1, 'height',
		5.0);
		pro.ele1.direction = -1;
		pro.ele2 =
		DummyElevator('processor',
			pro=elevatorProcessor;
			pro.ele1 =
			DummyElevator('processor',
			pro, 'floor', 2, 'id', 1, 'height',
			5.0);
			pro.ele1.direction = 0;
			pro.ele2 =
			DummyElevator('processor',

	pro, 'floor', 1, 'id', 1, 'height', 0.0); pro.ele2.direction = 1;	pro, 'floor', 1, 'id', 1, 'height', 0.0); pro.ele2.direction = -1;	pro, 'floor', 1, 'id', 1, 'height', 0.0); pro.ele2.direction = 0;
Expected Output	floorUI.currentFloor_1.Text==num2str(pro.ele1.floor) floorUI.currentFloor_2.Text==num2str(pro.ele2.floor) eleUI.currentFloor.Text==num2str(pro.ele2.floor) floorUI.upLamp_1.Color==[0,1,0] floorUI.upLamp_2.Color==[0,1,0] eleUI.upLamp.Color==[0,1,0]	floorUI.downLamp_1.Color==[0,1,0] floorUI.downLamp_2.Color==[0,1,0] eleUI.downLamp.Color==[0,1,0]	floorUI.upLamp_1.Color==[0,0,0] floorUI.upLamp_2.Color==[0,0,0] eleUI.upLamp.Color==[0,0,0] floorUI.downLamp_1.Color==[0,0,0] floorUI.downLamp_2.Color==[0,0,0] eleUI.downLamp.Color==[0,0,0]

- Test coverage: 12/12=100%
- Test result: 3 passed

T1.1.13 Test displayTime()

```
function displayTime(pro)
    %Method to refresh the clock in all UIs
    time=datetime('now');           % Statement - Tcover1.1.13.1
    time = string(datetime(time,'Format','HH:mm:ss'));
    pro.eleUIs(1).time.Text=time;
    pro.eleUIs(2).time.Text=time;
    pro.floorUIs(1).time.Text=time;
    pro.floorUIs(2).time.Text=time;
    pro.floorUIs(3).time.Text=time;
end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.1.13.1
Coverage Item	Tcover1.1.13.1
Input	--
State	pro=elevatorProcessor;
Expected Output	floorUI.time.Text==eleUI.time.Text

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

T1.2 InternalUI Unit Test

T1.2.1 Test Button Press

```
% Button pushed function: closeDoor
function closeDoorButtonPushed(app, event) % Statement – Tcover1.2.1.1
    app.closeDoor.BackgroundColor(3)=1-app.closeDoor.BackgroundColor(3);
end
```

```
% Button pushed function: openDoor
function openDoorButtonPushed(app, event) % Statement – Tcover1.2.1.2
    app.openDoor.BackgroundColor(3)=1-app.openDoor.BackgroundColor(3);
end


% Button pushed function: detectObj
function detectObjButtonPushed(app, event) % Statement – Tcover1.2.1.3
    app.detectObj.BackgroundColor(2)=1-app.detectObj.BackgroundColor(2);
    app.detectObj.BackgroundColor(3)=1-app.detectObj.BackgroundColor(3);
end
```

- Coverage Criteria: Branch coverage
- Test case

	Test Case T1.2.1.1
Coverage Item	Tcover1.2.1.1, Tcover1.2.1.2, Tcover1.2.1.3
Input	Press open door, close door, detectObj button
State	testCase.eleUI=internalUI;
Expected Output	sum(abs(testCase.eleUI.closeDoor.BackgroundColor-[0.96,0.96,0.04]))<0.01 sum(abs(testCase.eleUI.openDoor.BackgroundColor-[0.96,0.96,0.04]))<0.01 sum(abs(testCase.eleUI.detectObj.BackgroundColor-[0.96,0.04,0.04]))<0.01

- Test coverage: 3/3=100%
- Test result: 1 passed

T2: Integration Test

T2.1: elevatorProcessor+2InteralUI Integration

T2.1.1: Test Open Door Button

```
function testOpenDoorButton(testCase)
    % T2.1.1: Keep the door open when it is about to close
    testCase.press(testCase.ele1UI.openDoor);
    pause(4);
    testCase.verifyEqual(abs(testCase.pro.ele1.doorDis-1)<0.1, true);
    testCase.press(testCase.ele1UI.openDoor);
    pause(7);
    testCase.verifyEqual(abs(testCase.pro.ele1.doorDis,0)<0.1, true);
end
```

- Test case

	Test Case T2.1.1.1
Coverage Item	Tcover1.1.5, Tcover1.1.12
Input	Press open door button
State	Elevator 1 door closed
Expected Output	Elevator 1 door opens, then after button released, door closed

- Test coverage: 2/2=100%
- Test result: 1 passed

T2.1.2: Test Object Detected

```
function testObjectDetected(testCase)
    % T2.1.2: Detected object
    testCase.press(testCase.ele1UI.detectObj);
    pause(4);
    testCase.verifyEqual(abs(testCase.pro.ele1.doorDis-1)<0.1, true);
    testCase.press(testCase.ele1UI.detectObj);
    pause(7);
    testCase.verifyEqual(abs(testCase.pro.ele1.doorDis-0)<0.1, true);
end
```

- Test case

	Test Case T2.1.2.1
Coverage Item	Tcover1.1.5, Tcover1.1.12
Input	Press detect object button
State	Elevator 1 door closed
Expected Output	Elevator 1 door opens, then after object removed, door closed

- Test coverage: 2/2=100%
- Test result: 1 passed

T2.1.3: Test Internal Floor Button

```
function testInsidePress(testCase)
    % T2.1.3: Inside Press
    testCase.press(testCase.ele1UI.floor3);
    testCase.verifyEqual(sum(abs(testCase.ele1UI.floor3.BackgroundColor-
[1,1,0]))<0.01,true);
    testCase.verifyEqual(sum(abs(testCase.ele1UI.upLamp.Color-
[0,1,0]))<0.01,true);
    pause(2);
    testCase.press(testCase.ele1UI.floor1);
    testCase.verifyEqual(sum(abs(testCase.ele1UI.floor1.BackgroundColor-
[1,1,1]))<0.01,true);
    testCase.verifyEqual(sum(abs(testCase.ele1UI.downLamp.Color-
[0,0,0]))<0.01,true);
    pause(2);
end
```

- Test case

	Test Case T2.1.3.1
Coverage Item	Tcover1.1.2, Tcover1.1.8, Tcover1.1.9
Input	Press elevator 1 floor 3 button, then press elevator 1 floor 1 button
State	Elevator 1 stops at floor 1
Expected Output	Elevator 1 goes up, and ignore the requirement to go to floor 1

- Test coverage: 3/3=100%

- Test result: 1 passed

T2.1.4: Test Elevator Move

```
function testMove(testCase)
    % T2.1.4: Elevator move
    testCase.press(testCase.ele1UI.floor2);
    testCase.verifyEqual(sum(abs(testCase.ele1UI.floor2.BackgroundColor-
[1,1,0]))<0.01,true);
    pause(10);
    testCase.verifyEqual(abs(testCase.pro.ele1.height-5)<0.1,true);
end
```

- Test case

	Test Case T2.1.4.1
Coverage Item	Tcover1.1.2, Tcover1.1.3, Tcover1.1.4
Input	Press elevator 1 floor 2 button
State	Elevator 1 stops at floor 1
Expected Output	Elevator 1 goes up, and stops at floor 2

- Test coverage: 3/3=100%
- Test result: 1 passed

T2.1.5: Test Close Door Button

```
function testCloseDoorButton(testCase)
    % T2.1.5: Close door button
    testCase.press(testCase.ele2UI.openDoor);
    pause(4);
    testCase.verifyEqual(abs(testCase.pro.ele2.doorDis-1)<0.1, true);
    testCase.press(testCase.ele2UI.openDoor);
    testCase.press(testCase.ele2UI.closeDoor);
    pause(1);
    testCase.verifyEqual(abs(testCase.pro.ele2.doorDis-1)>0.1, true);
    testCase.press(testCase.ele2UI.closeDoor);
end
```

- Test case

	Test Case T2.1.5.1
Coverage Item	Tcover1.1.6, Tcover1.1.12
Input	Press elevator 2 close door button
State	Elevator 2 door opened
Expected Output	Elevator 2 door closes

- Test coverage: 2/2=100%
- Test result: 1 passed

T2.2: elevatorProcessor+2InteralUI+3ExteralUI Integration

This is a comprehensive test, consists of 6 subtests. The testing is continuous and no re-startup happens between 2 subtests. (Also named Demo Test)

```
function testDemoProcess(testCase)
```

```
    % This testcase is only used for demo.

    % T2.2.1: Outside Press
    testCase.press(testCase.floor2UI.down);
    testCase.verifyEqual(testCase.ele1UI.upLamp.Color,[0,1,0]);

    % T2.2.2: Keep the door open when it is about to close
    while testCase.ele1UI.doorDistance.Value < 0.5
        pause(0.5);
    end
    while testCase.ele1UI.doorDistance.Value >= 0.5
        pause(0.5);
    end
    testCase.press(testCase.ele1UI.openDoor);
    pause(4);
    testCase.verifyEqual(abs(testCase.pro.ele1.doorDis-1)<0.1, true);
    testCase.press(testCase.ele1UI.openDoor);

    % T2.2.3: Detected object
    while testCase.ele1UI.doorDistance.Value >= 0.5
        pause(0.5);
    end
    testCase.press(testCase.ele1UI.detectObj);
    pause(4);
    testCase.verifyEqual(abs(testCase.pro.ele1.doorDis-1)<0.1, true);
    testCase.press(testCase.ele1UI.detectObj);
    pause(1);

    % T2.2.4: Inside Press
    testCase.press(testCase.ele1UI.floor3);
    testCase.verifyEqual(sum(abs(testCase.ele1UI.floor3.BackgroundColor-
[1,1,1]))<0.01,true);
    pause(2);
    testCase.press(testCase.ele1UI.floor1);
    testCase.verifyEqual(sum(abs(testCase.ele1UI.floor1.BackgroundColor-
[1,1,0]))<0.01,true);

    % T2.2.5: Elevator management
    while testCase.ele1UI.height.Value > 3
        pause(0.5);
    end
    testCase.press(testCase.floor2UI.up);
    pause(0.5);
    testCase.verifyEqual(sum(abs(testCase.ele2UI.downLamp.Color-
[0,1,0]))<0.01,true);

    % T2.2.6: Close door button
    while testCase.ele2UI.doorDistance.Value <= 0.9
        pause(0.5);
    end
    testCase.press(testCase.ele2UI.closeDoor);
    pause(1);
    testCase.press(testCase.ele2UI.closeDoor);
    pause(4);

end
```

- Test case

	Test Case T2.2.1-T2.2.6
Coverage Item	Tcover1.1.1-Tcover1.1.13
Input	1. Press down button on floor 2 from outside; 2. When the door is about to close, press open door button; 3. Object detected; 4. Press floor 3 in elevator 1, and floor 1 in elevator 1; 5. Press up button on floor 2 from outside; 6. Press close door button of elevator 2 when door is open.
State	Elevator 1 on floor 1, elevator 2 on floor 3
Expected Output	2. The door of elevator 1 opens as wished; 3. The door of elevator 1 opens until object removed; 4. Press floor 3 makes no sense, and elevator responses to floor 1 press and start going down; 5. Elevator 2 responses to go down and stops at floor 2; 6. Elevator 2 closes the door.

- Test result: 1 passed

T3: Functional Test

T3.1: Use Case "Open Elevator Door"

T3.1.1: Static Press

- Test case

	Test Case T3.1.1
State	Elevator 1 on floor 1, elevator 2 on floor 3
Operation	1. Press "open door" button in two elevators; 2. Wait 4 seconds; 3. Release "open door" button in two elevators (Press again to simulate this); 4. Wait 6 seconds.
Expected Behavior	2. Elevators door open; 4. Elevators door close.

- Test result: 1 passed

T3.1.2: Reach Target Place

- Test case

	Test Case T3.1.2
State	Elevator 1 on floor 1, elevator 2 on floor 3
Operation	1. Press "Up" button on floor 2; 2. Wait elevator 1 up; 3. Wait 5 seconds;
Expected Behavior	3. Elevator 1 door open;

- Test result: 1 passed

T3.2: Use Case “Close Elevator Door”

T3.2.1: Static Press

- Test case

	Test Case T3.2.1.1	Test Case T3.2.1.2
State	Elevator 1 on floor 1, elevator 2 on floor 3	Elevator 1 on floor 1, elevator 2 on floor 3
Operation	1. Press “open door” button in two elevators; 2. Wait 4 seconds; 3. Release “open door” button in two elevators (Press again to simulate this); 4. Press “close door” button in two elevators;	1. Press “open door” button in two elevators; 2. Release “open door” button in two elevators (Press again to simulate this); 3. Press “close door” button in two elevators;
Expected Behavior	2. Elevators door open; 4. Elevators door close.	3. Elevators door open fully, then close;

- Test result: 2 passed

T3.2.2: Reach Target Place

- Test case

	Test Case T3.2.2
State	Elevator 1 on floor 1, elevator 2 on floor 3
Operation	1. Press “Up” button on floor 2; 2. Wait elevator 1 up;
Expected Behavior	2. Elevator 1 door keep closed;

- Test result: 1 passed

T3.3: Use Case “Elevator Detect Object”

T3.3.1: Detect Object

- Test case

	Test Case T3.3.1
State	Elevator 1 on floor 1, elevator 2 on floor 3
Operation	1. Press “open door” button in elevator 1; 2. Wait 4 seconds; 3. Release “open door” button in elevator 1 (Press again to simulate this); 4. Wait till the door close half. 5. Detect object (Press “detect object” button to simulate this);

	6. Wait 4 seconds; 7. Object removed (Press “detect object” button again to simulate this).
Expected Behavior	5. Elevators door open; 7. Elevators door close.

- Test result: 1 passed

T3.4: Use Case “Select Floor inside Elevator”

T3.4.1: Select Floor inside

- Test case

	Test Case T3.4.1
State	Elevator 1 on floor 1, elevator 2 on floor 3
Operation	1. Press floor 2 in elevator 1, floor 1 in elevator 2.
Expected Behavior	1. Elevator 1 goes to floor 2 eventually, elevator 2 goes to floor 1 eventually.

- Test result: 1 passed

T3.4.2: Select Multiple Floor

- Test case

	Test Case T3.4.2
State	Elevator 1 on floor 1, elevator 2 on floor 3
Operation	1. Press floor 2 and 3 in elevator 1.
Expected Behavior	1. Elevator 1 goes to floor 2 and 3 eventually.

- Test result: 1 passed

T3.4.3: Select Current Floor

- Test case

	Test Case T3.4.2
State	Elevator 1 on floor 1, elevator 2 on floor 3
Operation	1. Press floor 1 in elevator 1.
Expected Behavior	1. Elevator 1 opens the door.

- Test result: 1 passed

T3.5: Use Case “Cancel Floor inside Elevator”

T3.5.1 Cancel Floor

- Test case

	Test Case T3.5.1
State	Elevator 1 on floor 1, elevator 2 on floor 3

Operation	1. Press floor 2, 3 in elevator 1. 2. Press floor 2 in elevator 1. 3. Wait for elevator 1 opens the door.
Expected Behavior	3. Elevator 1 goes to floor 3 eventually.

- Test result: 1 passed

T3.6: Use Case "Display Info Inside and Outside"

T3.6.1 Inside Floor Button

- Test case

	Test Case T3.6.1
State	Elevator 1 on floor 1, elevator 2 on floor 3
Operation	1. Press floor 2 in elevator 1. 2. Wait for elevator 1 opens the door.
Expected Behavior	1. Elevator 1 floor 2 lights on. 2. Elevator 1 floor 2 lights off.

- Test result: 1 passed

T3.6.2 Inside Door Button

- Test case

	Test Case T3.6.2
State	Elevator 1 on floor 1, elevator 2 on floor 3
Operation	1. Press "open door" button, "close door" button in elevator 1.
Expected Behavior	1. Elevator 1 "open door" button, "close door" button lights on.

- Test result: 1 passed

T3.6.3 Floor Number

- Test case

	Test Case T3.6.3
State	Elevator 1 on floor 1, elevator 2 on floor 3
Operation	1. Press floor 3 in elevator 1, floor 2 in elevator 2; 2. Wait till they are ready.
Expected Behavior	1. Elevator 1 show floor 3, elevator 2 show floor 2, all outside UI show elevator 1 on floor 3 and elevator 2 on floor 2.

- Test result: 1 passed

T3.6.4 Outside Button

- Test case

	Test Case T3.6.4

State	Elevator 1 on floor 1, elevator 2 on floor 3
Operation	1. Press down on floor 2. 2. Wait for elevator 1 opens the door.
Expected Behavior	1. Floor 2 down button lights on. 2. Floor 2 down button lights off.

- Test result: 1 passed

T3.6.5 Up Down Lamp

- Test case

	Test Case T3.6.5
State	Elevator 1 on floor 1, elevator 2 on floor 3
Operation	1. Press down on floor 3. Press floor 2 in elevator 1.
Expected Behavior	1. Elevator 1 up lights on, elevator 2 down lights on, all outside UI show elevator 1 up lights and elevator 2 down lights.

- Test result: 1 passed

T3.7: Use Case "Call Elevator Outside"

- Test case

	Test Case T3.7.1
State	Elevator 1 on floor 1, elevator 2 on floor 3
Operation	1. Press down on floor 2.
Expected Behavior	1. Elevator 1 goes to floor 2 eventually.

- Test result: 1 passed

T3.8: Elevator Task Management

T3.8.1: Multi-calls with directions

- Test case

	Test Case T3.8.1
State	Elevator 1 on floor 1, elevator 2 on floor 3
Operation	1. Press up and down on floor 2.
Expected Behavior	1. Elevator 1 and 2 goes to floor 2 eventually, with elevator 1 goes up, elevator 3 goes down.

- Test result: 1 passed

T3.8.2: Convenience

- Test case

	Test Case T3.8.2
State	Elevator 1 on floor 1, elevator 2 on floor 3

Operation	1. Press floor 3 in elevator 1; 2. Press up on floor 2.
Expected Behavior	2. Elevator 1 goes to floor 2, then floor 3.

- Test result: 1 passed

Model Checking

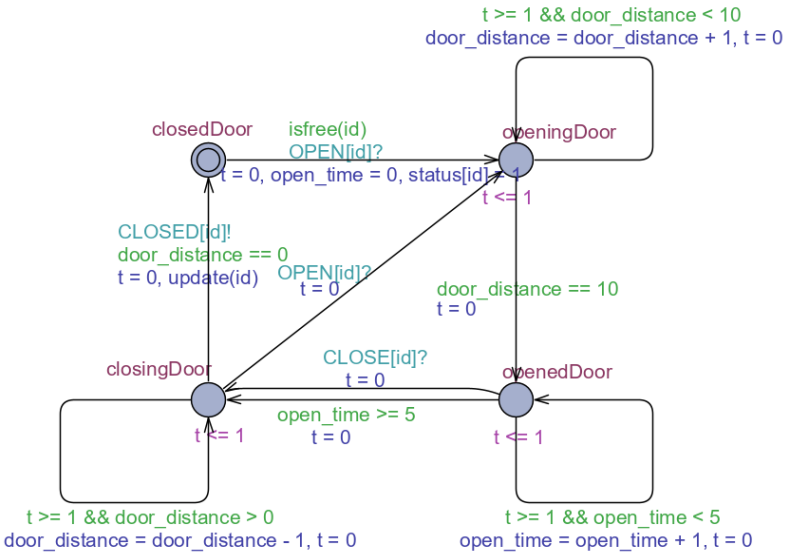
A UPPAAL model of this elevator system is built for model checking.

Full Elevator Model

The full UPPAAL model consists of 4 parts: 1. The elevator door template; 2. The elevator template; 3. The user template; 4. The processor template.

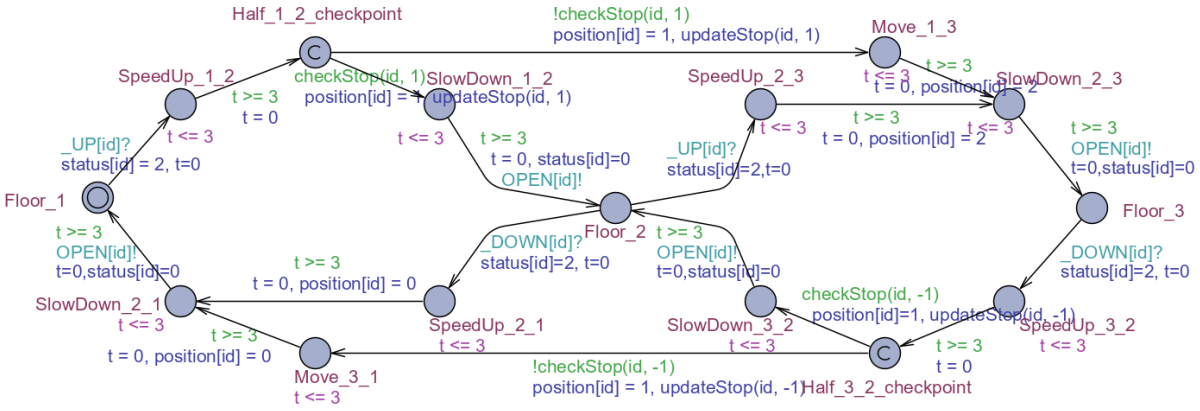
The full model consists of 2 elevator doors, 2 elevators, 1 user and 1 processor.

The elevator door



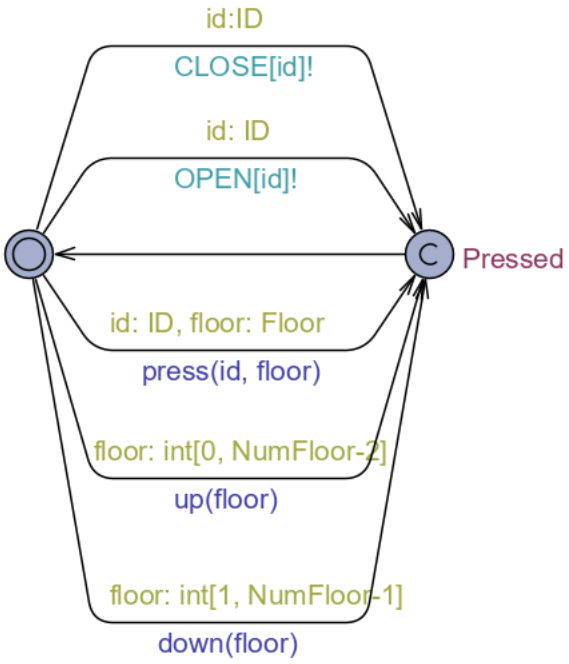
This simulates the process of the door of an elevator. It will open the door depend on requests. Open door request is prior to close door request. The door will automatically close in 5 seconds if no open door request received.

The elevator



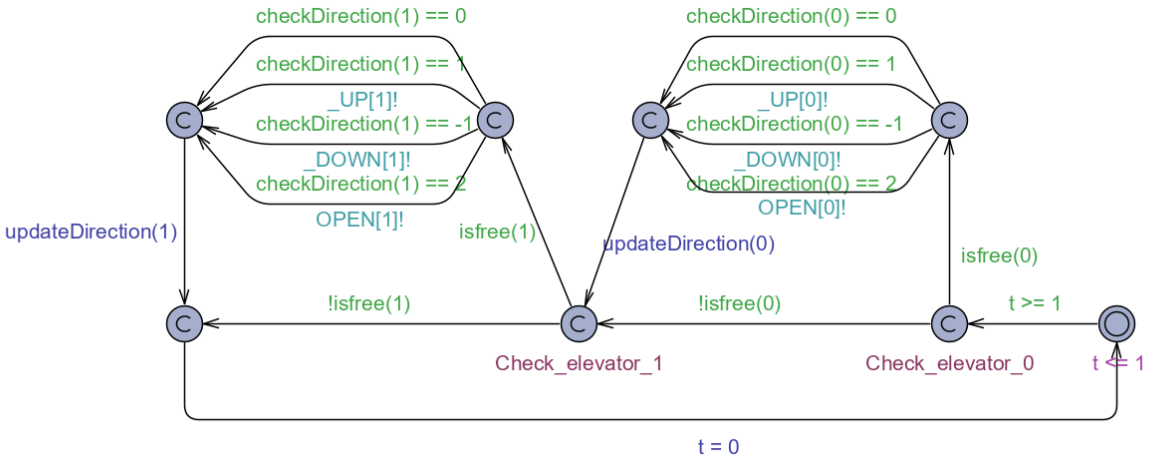
This model simulates the elevator object. Elevator moves between 3 floors, and speed up and slow down to move the elevator. If request comes in when elevator moving, it will check if it is able to take it temporarily, or just pass the middle floors, at the half checkpoints.

The User



User will be able to press any button at any time.

The Processor



The processor will check the elevator status every 1 second (in implementation, 0.1 sec), and distribution tasks to elevators.

Data storage

1. internalOrder[i * NumFloor + j]: Pressed ‘j+1’ floor button on elevator ‘i+1’. NumFloor = 3.
2. externalOrder[i * NumFloor + j]: If i=0, press up button outside on floor ‘j+1’; If i=1, press down button outside on floor ‘j+1’.

Check Properties

Since the whole model is too large that only existence property can be checked without running out of memory, the properties checked is very limited:

P1.1

Property	E<> ElevatorDoor(0).openedDoor
Description	The elevator will open the door sometime.
Result	Passed

P1.2

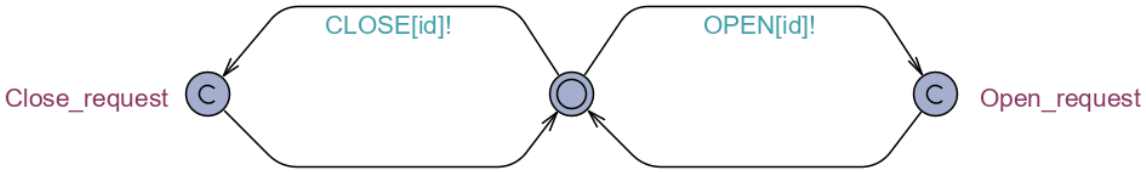
Property	E<> Elevator(0).Floor_3
Description	Elevator 1 is able to get to floor 3
Result	Passed

Since the model still need to be checked, several sub models that only involves parts of the system is built for model checking.

Sub Door Model

This model only consists 2 elevator doors and 2 users, each user controls one elevator door.

The elevator door model is the same as the full model. The user model changes a little bit:



Such change does not affect the environment assumption, but better for model checking.

Below are the properties checked:

P2.1

Property	A[] not deadlock
Description	The system will not crash and has no deadlock.
Result	Passed

P2.2

Property	A[] ElevatorDoor(0).door_distance >= 0 && ElevatorDoor(0).door_distance <= 10
Description	The door will never closed too much or open too wide
Result	Passed

P2.3

Property	E<> ElevatorDoor(0).openedDoor
Description	The elevator will open the door sometime.
Result	Passed

P2.4

Property	A[] ElevatorDoor(0).open_time <= 5

Description	The elevator won't be open for longer than 5 seconds -- otherwise it will always try to close the door.
Result	Passed

P2.5

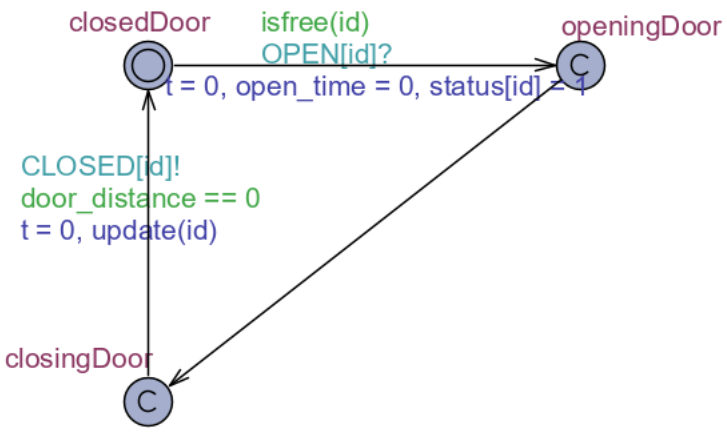
Property	A[] forall (i: ID) User(i).Close_request imply not ElevatorDoor(i).openedDoor
Description	If the user presses the closing door button (opening door button will not be pressed), then the door will not keep opened.
Result	Passed

P2.6

Property	A[] forall (i: ID) User(i).Open_request imply not ElevatorDoor(i).closingDoor and not ElevatorDoor(i).closedDoor
Description	If the user presses the opening door button, then the door will try to open (not closing nor closed).
Result	Passed

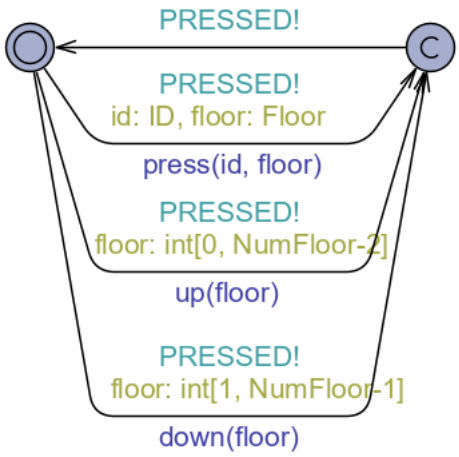
Sub Elevator Model

The elevator model is mostly same as the full model, but simplifies the door model:



The door open and close event will be finished in one moment.

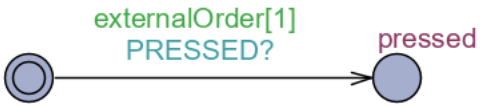
User model is simplified a little (unable to open and close door):



And each move will send a broadcast for easier model checking.

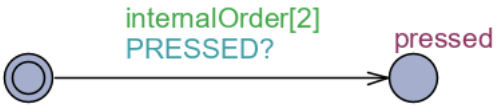
Besides, 2 helper template is built for model checking:

Helper_1:



State pressed will be reached if and only if user pressed some button after press up button on floor 2.

Helper_2:



State pressed will be reached if and only if user pressed some button after press floor 3 in elevator 1.

P3.1

Property	A[] not deadlock
Description	The system will not crash and has no deadlock.
Result	Passed

P3.2

Property	A[] forall (i:ID) ElevatorDoor(i).openingDoor imply (Elevator(i).Floor_1 or Elevator(i).Floor_2 or Elevator(i).Floor_3)
Description	Whenever the elevator door is open(ing), the elevator is stopped on some floor (not moving).
Result	Passed

P3.3

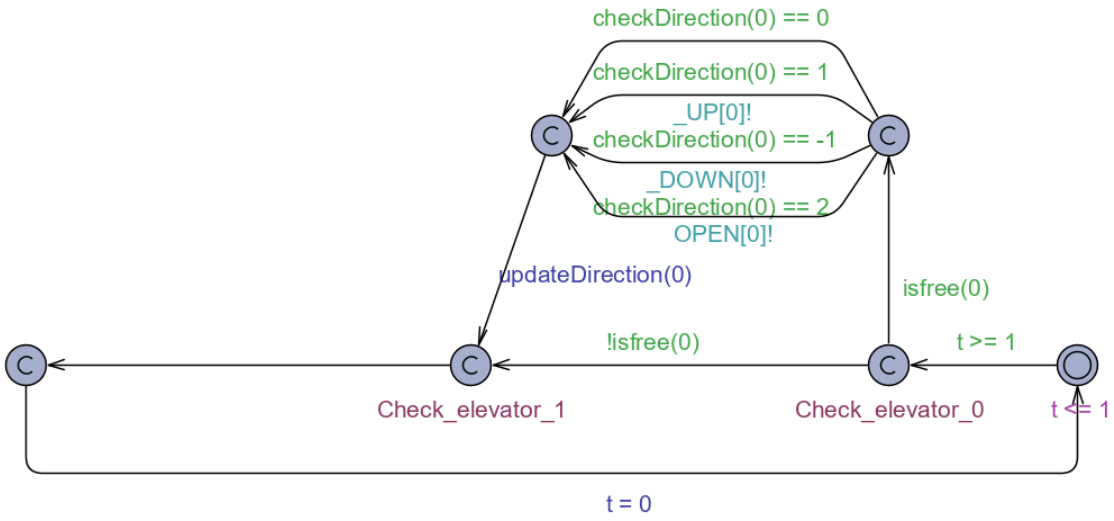
Property	externalOrder[1] --> (Elevator(1).Floor_2 or Elevator(0).Floor_2 or Helper_1.pressed)
Description	Once up button on floor 2 pressed, 3 possibilities: 1. Elevator 1 goes to floor 2; 2. Elevator 2 goes to floor 2; 3. The user keeps press button to keep the elevator on other floors (e.g. both elevator on floor 3 and users keep press floor 3 inside elevators)
Result	Passed

P3.4

Property	internalOrder[2] --> (Elevator(0).Floor_3 or Helper_2.pressed)
Description	Once floor 3 button in elevator 1 pressed, 3 possibilities: 1. Elevator 1 goes to floor 3; 2. The user keeps press button to keep the elevator on other floors (e.g. elevator 1 on floor 1 and users keep press floor 1 inside elevator)
Result	Passed

Single Elevator Model

This model only contains one elevator door and one elevator. The processor need adjustment:



Just delete the elevator 1 part totally.

Below are the properties checked:

P4.1

Property	A[] not deadlock
Description	The system will not crash and has no deadlock.
Result	Passed

P4.2

Property	internalOrder[0] --> Elevator.Floor_1 or User.Pressed
Description	This test describes that if a request in elevator 1 floor 1 applies, elevator will 1. eventually goes to floor 1, or 2. another user pressed some button later on (e.g. keep pressing open door button to prevent the elevator go to floor 1.)
Result	Passed

P4.3

Property	internalOrder[1] --> Elevator.Floor_2 or User.Pressed
Description	This test describes that if a request in elevator 1 floor 2 applies, elevator will

	1. eventually goes to floor 2, or 2. another user pressed some button later on (e.g. keep pressing open door button to prevent the elevator go to floor 2.)
Result	Passed

P4.4

Property	internalOrder[2] --> Elevator.Floor_3 or User.Pressed
Description	This test describes that if a request in elevator 1 floor 3 applies, elevator will 1. eventually goes to floor 3, or 2. another user pressed some button later on (e.g. keep pressing open door button to prevent the elevator go to floor 3.)
Result	Passed

P4.5

Property	internalOrder[2] --> Elevator.Floor_3 or User.Pressed
Description	This test describes that if a request in elevator 1 floor 3 applies, elevator will 1. eventually goes to floor 3, or 2. another user pressed some button later on (e.g. keep pressing open door button to prevent the elevator go to floor 3.)
Result	Passed

P4.6

Property	externalOrder[1*NumFloor+2] --> Elevator.Floor_3 or User.Pressed
Description	This test describes that if a request on floor 3 with down direction applies, elevator will 1. eventually goes to floor 3, or 2. another user pressed some button later on (e.g. keep pressing open door button to prevent the elevator go to floor 3.)
Result	Passed

P4.7

Property	externalOrder[1*NumFloor+1] --> Elevator.Floor_2 or User.Pressed
Description	This test describes that if a request on floor 2 with down direction applies, elevator will 1. eventually goes to floor 2, or 2. another user pressed some button later on (e.g. keep pressing open door button to prevent the elevator go to floor 2.)
Result	Passed

P4.8

Property	externalOrder[0*NumFloor+1] --> Elevator.Floor_2 or User.Pressed
Description	This test describes that if a request on floor 2 with up direction applies, elevator will 1. eventually goes to floor 2, or 2. another user pressed some button later on (e.g. keep pressing open door button to prevent the elevator go to floor 2.)
Result	Passed

P4.9

Property	externalOrder[0*NumFloor+0] --> Elevator.Floor_1 or User.Pressed
Description	This test describes that if a request on floor 1 with up direction applies, elevator will 1. eventually goes to floor 1, or 2. another user pressed some button later on (e.g. keep pressing open door button to prevent the elevator go to floor 1.)
Result	Passed

P4.10

Property	A[] not ElevatorDoor.closedDoor imply (Elevator.Floor_1 or Elevator.Floor_2 or Elevator.Floor_3)
Description	Whenever the elevator door is not closed, the elevator is stopped on some floor (not moving).
Result	Passed