SOFTWARE VALIDATIONS

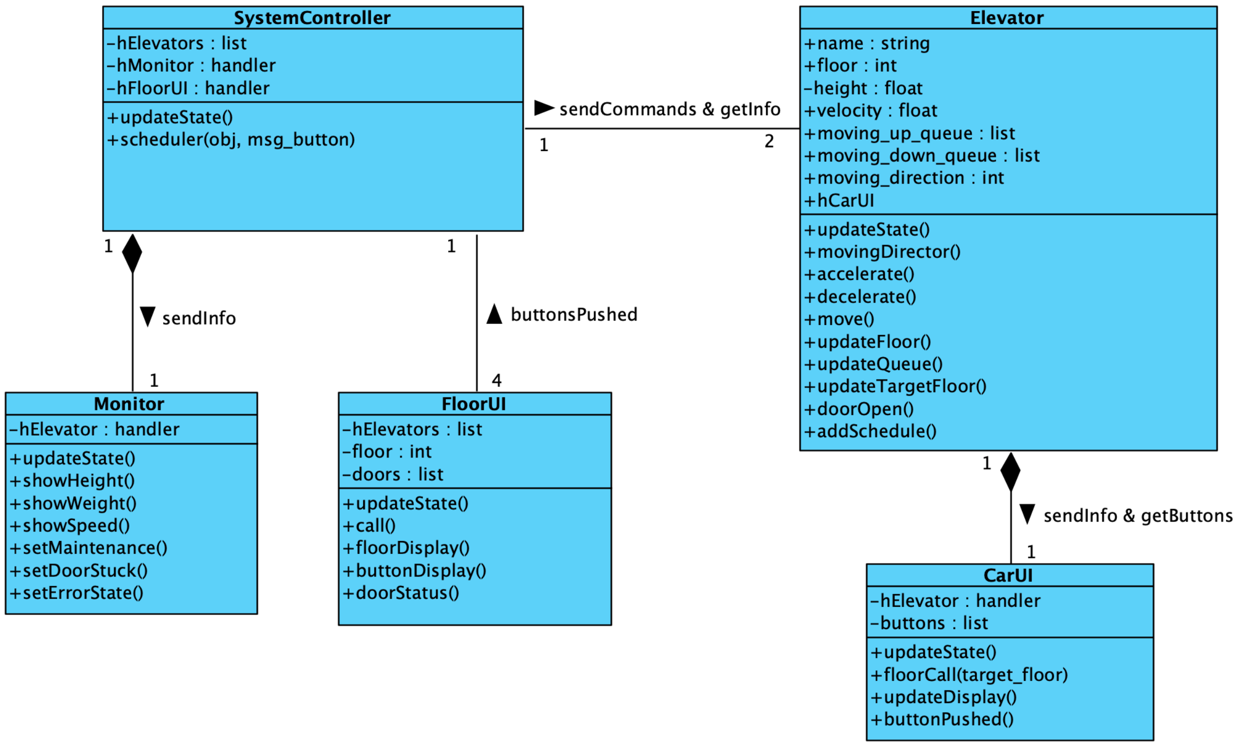
Elevator System

Team 1
Author: Ziqi Gao

Table of Contents

System Architecture

The elevator system is controlled by a *SystemController*, receiving button information from each *FloorUI* and *Elevator*, and sending information and commands to the *Monitor* and *Elevator*. Each Elevator has a CarUI, with which user can observe the elevator status and select floors to go. The architecture of the elevator system is shown below.



T1: Unit Test

Note: TCover1.1.1.1.1 means that Branch – Tcover1.1.1.1 takes True, and TCover1.1.1.1.2 means that Branch – Tcover1.1.1.1 takes False. All the names following are defined in this way.

T1.1: SystemController Unit Test

T1.1.1: Test buildHierarchy()

```
function [elevator_hierarchy] = buildHierarchy(obj, floor_calling, direction_calling)
    elevator_hierarchy = [0,0];
    % Build hierarchy
    for i = 1:2
        if obj.hElevators(i).moving_direction == 1 ||
obj.hElevators(i).direction_store == 1    % Branch - Tcover1.1.1.1
            if floor_calling > obj.hElevators(i).height    % Branch - Tcover1.1.1.2
                if direction_calling == "up"    % Branch - Tcover1.1.1.3
                    elevator_hierarchy(i) = 1;
                elseif direction_calling == "down"    % Branch - Tcover1.1.1.4
                    elevator_hierarchy(i) = 2;
                end
            elseif floor_calling <= obj.hElevators(i).height  % Branch - Tcover1.1.1.5
                if direction_calling == "up"    % Branch - Tcover1.1.1.6
                    elevator_hierarchy(i) = 4;
                elseif direction_calling == "down"    % Branch - Tcover1.1.1.7
                    elevator_hierarchy(i) = 3;
                end
            end
        elseif obj.hElevators(i).moving_direction == -1 ||
obj.hElevators(i).direction_store == -1    % Branch - Tcover1.1.1.8
            if floor_calling > obj.hElevators(i).height    % Branch - Tcover1.1.1.9
                if direction_calling == "up"    % Branch - Tcover1.1.1.10
                    elevator_hierarchy(i) = 3;
                elseif direction_calling == "down"    % Branch - Tcover1.1.1.11
                    elevator_hierarchy(i) = 4;
                end
            elseif floor_calling <= obj.hElevators(i).height % Branch - Tcover1.1.1.12
                if direction_calling == "up"    % Branch - Tcover1.1.1.13
                    elevator_hierarchy(i) = 2;
                elseif direction_calling == "down"    % Branch - Tcover1.1.1.14
                    elevator_hierarchy(i) = 1;
                end
            end
        % If still, hierarchy=0
        end
    end

    % Special case: go to basement
    if floor_calling == 1 && direction_calling == "down"    % Branch - Tcover1.1.1.13
        elevator_hierarchy(2) = -1;
    end
    % Special case: basement call
    if floor_calling == 0    % Branch - Tcover1.1.1.14
        elevator_hierarchy(2) = -1;
    end

end
```

- Coverage Criteria: Branch coverage
- Test case

Test Case	Test Case T1.1.1.1	Test Case T1.1.1.2	Test Case T1.1.1.3	Test Case T1.1.1.4
Coverage Item	TCover1.1.1.1.1, TCover1.1.1.2.1, TCover1.1.1.3.1, TCover1.1.1.13.2, TCover1.1.1.14.2	TCover1.1.1.1.1, TCover1.1.1.2.1, TCover1.1.1.3.2, TCover1.1.1.4.1, TCover1.1.1.13.2, TCover1.1.1.14.2	TCover1.1.1.1.1, TCover1.1.1.2.2, TCover1.1.1.5.1, TCover1.1.1.6.1, TCover1.1.1.13.2, TCover1.1.1.14.2	TCover1.1.1.1.1, TCover1.1.1.2.2, TCover1.1.1.5.1, TCover1.1.1.6.2, TCover1.1.1.7.1, TCover1.1.1.13.2, TCover1.1.1.14.2
Input	[2, ‘up’]	[3, ‘down’]	[1, ‘up’]	[2, ‘down’]
State	Elevator.moving_direction=1; Elevator.height=1.3;	Elevator.moving_direction=1; Elevator.height=1.3;	Elevator.moving_direction=1; Elevator.height=2.3;	Elevator.moving_direction=1; Elevator.height=2.3;
Expected Output	Elevator_hierarchy==1	Elevator_hierarchy==2	Elevator_hierarchy==4	Elevator_hierarchy==3
Test Case	Test Case T1.1.1.5	Test Case T1.1.1.6	Test Case T1.1.1.7	Test Case T1.1.1.8
Coverage Item	TCover1.1.1.1.2, TCover1.1.1.8.1, TCover1.1.1.9.1, TCover1.1.1.10.1, TCover1.1.1.13.2, TCover1.1.1.14.2	TCover1.1.1.1.2, TCover1.1.1.8.1, TCover1.1.1.9.1, TCover1.1.1.10.2, TCover1.1.1.11.1, TCover1.1.1.13.2,	TCover1.1.1.1.2, TCover1.1.1.8.1, TCover1.1.1.9.2, TCover1.1.1.12.1, TCover1.1.1.13.1, TCover1.1.1.13.2,	TCover1.1.1.1.2, TCover1.1.1.8.1, TCover1.1.1.9.2, TCover1.1.1.12.1, TCover1.1.1.13.2, TCover1.1.1.14.1,

		TCover1.1.1.14.2	TCover1.1.1.14.2	TCover1.1.1.13.2, TCover1.1.1.14.2
Input	[2, ‘up’]	[3, ‘down’]	[1, ‘up’]	[2, ‘down’]
State	Elevator.moving_direction=-1; Elevator.height=1.3;	Elevator.moving_direction=-1; Elevator.height=1.3;	Elevator.moving_direction=-1; Elevator.height=2.3;	Elevator.moving_direction=-1; Elevator.height=2.3;
Expected Output	Elevator_hierarchy==3	Elevator_hierarchy==4	Elevator_hierarchy==2	Elevator_hierarchy==1
Test Case	Test Case T1.1.1.9	Test Case T1.1.1.10		
Coverage Item	TCover1.1.1.1.1, TCover1.1.1.2.2, TCover1.1.1.5.1, TCover1.1.1.6.1, TCover1.1.1.13.1, TCover1.1.1.14.2	TCover1.1.1.1.1, TCover1.1.1.2.2, TCover1.1.1.5.1, TCover1.1.1.6.2, TCover1.1.1.7.1, TCover1.1.1.13.2, TCover1.1.1.14.1		
Input	[0, ‘up’]	[1, ‘down’]		
State	Elevator2.moving_direction=1; Elevator2.height=1.3;	Elevator2.moving_direction=1; Elevator2.height=1.3;		
Expected Output	Elevator2_hierarchy==-1	Elevator2_hierarchy==-1		

- Test coverage: 28/28 = 100%
- Test result: 10 passed

T1.1.2: Test scheduler()

```
function scheduler(obj, msg_button)
% Elevator dispatching
    direction_calling = msg_button(1);
    floor_calling = str2double(msg_button(2));

    % Build hierarchy
    elevator_hierarchy = obj.buildHierarchy(floor_calling, direction_calling);

    % Compare hierarchy
    if elevator_hierarchy(1) < elevator_hierarchy(2)    % Branch - Tcover1.1.2.1
        obj.hElevators(1).addSchedule(floor_calling, direction_calling)
    elseif elevator_hierarchy(1) > elevator_hierarchy(2)    % Branch - Tcover1.1.2.2
        obj.hElevators(2).addSchedule(floor_calling, direction_calling)
    else    % Branch - Tcover1.1.2.3
        dist1 = abs(obj.hElevators(1).height-floor_calling);
        dist2 = abs(obj.hElevators(2).height-floor_calling);
        if dist1 <= dist2    % Branch - Tcover1.1.2.4
            obj.hElevators(1).addSchedule(floor_calling, direction_calling)
        else    % Branch - Tcover1.1.2.5
            obj.hElevators(2).addSchedule(floor_calling, direction_calling)
        end
    end

end
```

- Coverage Criteria: Branch coverage
- Test case

Test Case	Test Case T1.1.2.1	Test Case T1.1.2.2	Test Case T1.1.2.3	Test Case T1.1.2.4
Coverage Item	TCover1.1.2.1.1	TCover1.1.2.1.2, TCover1.1.2.2.1	TCover1.1.2.1.2, TCover1.1.2.2.2, TCover1.1.2.3.1, TCover1.1.2.4.1	TCover1.1.2.1.2, TCover1.1.2.2.2, TCover1.1.2.3.1, TCover1.1.2.4.2, TCover1.1.2.5.1
Input	[2, ‘up’]	[3, ‘down’]	[1, ‘up’]	[2, ‘down’]
State	elevator_hierarchy = [0, 1]	elevator_hierarchy = [1, 0]	elevator_hierarchy = [0, 0]; dist1=1, dist2=2	elevator_hierarchy = [0, 0]; dist1=2, dist2=1
Expected Output	Elevator1.addSchedule()	Elevator2.addSchedule()	Elevator1.addSchedule()	Elevator2.addSchedule()

- Test coverage: 8/8 = 100%
- Test result: 4 passed

T1.2: Elevator Test

T1.2.1: Test movingDirector()

```
function movingDirector(obj)
    % If moving up
    flag1 = 0;
    flag2 = 0;
    if obj.moving_direction == 1     % Branch - Tcover1.2.1.1
        if isempty(obj.moving_up_queue) == 0     % Branch - Tcover1.2.1.2
            if obj.floor < obj.moving_up_queue(end) % Branch - Tcover1.2.1.3
                flag1 = 1;
            end
        end
        if isempty(obj.moving_down_queue) == 0  % Branch - Tcover1.2.1.4
            if obj.floor < obj.moving_down_queue(1) % Branch - Tcover1.2.1.5
                flag2 = 1;
            end
        end
    elseif obj.moving_direction == -1    % Branch - Tcover1.2.1.6
        if isempty(obj.moving_up_queue) == 0     % Branch - Tcover1.2.1.7
            if obj.floor > obj.moving_up_queue(1)    % Branch - Tcover1.2.1.8
                flag1 = 1;
            end
        end
        if isempty(obj.moving_down_queue) == 0  % Branch - Tcover1.2.1.9
            if obj.floor > obj.moving_down_queue(end)    % Branch - Tcover1.2.1.10
                flag2 = 1;
            end
        end
    else  % still, Redirecting  % Branch - Tcover1.2.1.11
        if isempty(obj.moving_up_queue) == 0     % Branch - Tcover1.2.1.12
            if obj.floor > obj.moving_up_queue(1)    % Branch - Tcover1.2.1.13
                obj.moving_direction = -1;
                flag1 = 1;
            elseif obj.floor < obj.moving_up_queue(1)    % Branch - Tcover1.2.1.14
                obj.moving_direction = 1;
                flag2 = 1;
            end
        elseif isempty(obj.moving_down_queue) == 0  % Branch - Tcover1.2.1.15
            if obj.floor < obj.moving_down_queue(1) % Branch - Tcover1.2.1.16
```

```
                obj.moving_direction = 1;
                flag1 = 1;
            elseif obj.floor > obj.moving_down_queue(1) % Branch - Tcover1.2.1.17
                obj.moving_direction = -1;
                flag2 = 1;
            end
        end
    end
    % Stop condition
    if flag1==0 && flag2==0 % Branch - Tcover1.2.1.18
        obj.moving_direction = 0;
    end
end
```

- Coverage Criteria: Branch coverage
- Test case

Test Case	Test Case T1.2.1.1	Test Case T1.2.1.2	Test Case T1.2.1.3	Test Case T1.2.1.4
Coverage Item	TCover1.2.1.1.1, TCover1.2.1.2.1, TCover1.2.1.3.1, TCover1.2.1.4.2, TCover1.2.1.19.2	TCover1.2.1.1.1, TCover1.2.1.2.1, TCover1.2.1.3.1, TCover1.2.1.4.1, TCover1.2.1.5.1, TCover1.2.1.19.1	TCover1.2.1.1.1, TCover1.2.1.2.2, TCover1.2.1.4.1, TCover1.2.1.5.2, TCover1.2.1.19.2	TCover1.2.1.1.1, TCover1.2.1.2.1, TCover1.2.1.3.2, TCover1.2.1.4.1, TCover1.2.1.5.1, TCover1.2.1.19.2
Input	—	—	—	—
State	moving_direction = 1; floor = 1; moving_up_queue = [2,3]; moving_down_queue = [];	moving_direction = 1; floor = 1; moving_up_queue = [2,3]; moving_down_queue = [2];	moving_direction = 1; floor = 1; moving_up_queue = []; moving_down_queue = [3,2];	moving_direction = 1; floor = 1; moving_up_queue = [0]; moving_down_queue = [2];
Expected Output	moving_direction=1	moving_direction=0	moving_direction=1	moving_direction=1
Test Case	Test Case T1.2.1.5	Test Case T1.2.1.6	Test Case T1.2.1.7	Test Case T1.2.1.8
Coverage Item	TCover1.2.1.1.2, TCover1.2.1.6.1, TCover1.2.1.7.1, TCover1.2.1.8.1, TCover1.2.1.9.2, TCover1.2.1.19.2	TCover1.2.1.1.2, TCover1.2.1.6.1, TCover1.2.1.7.1, TCover1.2.1.8.1, TCover1.2.1.9.1, TCover1.2.1.10.1, TCover1.2.1.19.1	TCover1.2.1.1.2, TCover1.2.1.6.1, TCover1.2.1.7.2, TCover1.2.1.8.1, TCover1.2.1.9.2, TCover1.2.1.19.2	TCover1.2.1.1.2, TCover1.2.1.6.1, TCover1.2.1.7.1, TCover1.2.1.8.2, TCover1.2.1.9.1, TCover1.2.1.10.1, TCover1.2.1.19.2
Input	—	—	—	—
State	moving_direction = -1; floor = 1; moving_up_queue = [0]; moving_down_queue = [];	moving_direction = -1; floor = 2; moving_up_queue = [3]; moving_down_queue = [2,1];	moving_direction = -1; floor = 2; moving_up_queue = []; moving_down_queue = [2,1];	moving_direction = -1; floor = 2; moving_up_queue = [3]; moving_down_queue = [2,1];
Expected Output	moving_direction=-1	moving_direction=0	moving_direction=-1	moving_direction=-1
Test Case	Test Case T1.2.1.9	Test Case T1.2.1.10	Test Case T1.2.1.11	Test Case T1.2.1.12

Coverage Item	TCover1.2.1.1.2, TCover1.2.1.6.2, TCover1.2.1.11.1, TCover1.2.1.12.1, TCover1.2.1.13.1	TCover1.2.1.1.2, TCover1.2.1.6.2, TCover1.2.1.11.1, TCover1.2.1.12.1, TCover1.2.1.13.2, TCover1.2.1.14.1	TCover1.2.1.1.2, TCover1.2.1.6.2, TCover1.2.1.11.1, TCover1.2.1.12.2, TCover1.2.1.15.1, TCover1.2.1.16.1	TCover1.2.1.1.2, TCover1.2.1.6.2, TCover1.2.1.11.1, TCover1.2.1.12.2, TCover1.2.1.15.1, TCover1.2.1.16.2, TCover1.2.1.17.1
Input	—	—	—	—
State	moving_direction = 0; floor = 3; moving_up_queue = [2,3]; moving_down_queue = [];	moving_direction = 0; floor = 1; moving_up_queue = [2,3]; moving_down_queue = [];	moving_direction = 0; floor = 1; moving_up_queue = []; moving_down_queue = [3,2];	moving_direction = 0; floor = 3; moving_up_queue = []; moving_down_queue = [2];
Expected Output	moving_direction=-1	moving_direction=1	moving_direction=1	moving_direction=-1
Test Case	Test Case T1.2.1.13	Test Case T1.2.1.14		
Coverage Item	TCover1.2.1.1.2, TCover1.2.1.6.2, TCover1.2.1.11.1, TCover1.2.1.12.1, TCover1.2.1.13.2, TCover1.2.1.14.2, TCover1.2.1.19.2	TCover1.2.1.1.2, TCover1.2.1.6.2, TCover1.2.1.11.1, TCover1.2.1.12.2, TCover1.2.1.15.1, TCover1.2.1.16.2, TCover1.2.1.17.2, TCover1.2.1.19.2		
Input	—	—		
State	moving_direction = 0; floor = 1; moving_up_queue = [1]; moving_down_queue = [];	moving_direction = 0; floor = 1; moving_up_queue = []; moving_down_queue = [1];		
Expected Output	moving_direction=0	moving_direction=0		

- Test coverage: 35/35 = 100%
- Test result: 14 passed

T1.2.2: Test Accelerate()

```
function Accelerate(obj)
    if obj.velocity < obj.velocity_max  % Branch - Tcover1.2.2.1
        obj.accelerate_status = 1;
        obj.velocity = obj.velocity + obj.accelerate;
    elseif obj.velocity > obj.velocity_max  % Branch - Tcover1.2.2.2
        obj.velocity = obj.velocity_max;
    end
    % Update accelerate status
    if obj.velocity >= obj.velocity_max  % Branch - Tcover1.2.2.3
        obj.accelerate_status = 0;
    end
end
```

- Coverage Criteria: Branch coverage
- Test case

Test Case	Test Case T1.2.2.1	Test Case T1.2.2.2	Test Case T1.2.2.3
Coverage Item	TCover1.2.2.1.1, TCover1.2.2.3.2	TCover1.2.2.1.2, TCover1.2.2.2.1, TCover1.2.2.3.1	TCover1.2.2.1.2, TCover1.2.2.2.2, TCover1.2.2.3.1
Input	—	—	—
State	velocity = 0.1	velocity = 0.3	velocity = 0.2
Expected Output	velocity += 0.1	velocity = 0.2	velocity = 0.2

- Test coverage: 6/6 = 100%
- Test result: 3 passed

T1.2.3: Test Decelerate()

```
function Decelerate(obj)
    if obj.velocity > 0  % Branch - Tcover1.2.3.1
        obj.accelerate_status = -1;
        obj.velocity = obj.velocity - obj.accelerate;
    elseif obj.velocity < 0  % Branch - Tcover1.2.3.2
        obj.velocity = 0;
    end
    % Update accelerate status
    if obj.velocity <= 0  % Branch - Tcover1.2.3.3
        obj.accelerate_status = 0;
    end
end
```

- Coverage Criteria: Branch coverage
- Test case

Test Case	Test Case T1.2.3.1	Test Case T1.2.3.2	Test Case T1.2.3.3
Coverage Item	TCover1.2.3.1.1, TCover1.2.3.3.2	TCover1.2.3.1.2, TCover1.2.3.2.1, TCover1.2.3.3.1	TCover1.2.3.1.2, TCover1.2.3.2.2, TCover1.2.3.3.1
Input	—	—	—
State	velocity = 0.1	velocity = -0.1	velocity = 0.0
Expected Output	velocity -= 0.1	velocity = 0.0	velocity = 0.0

- Test coverage: 6/6 = 100%
- Test result: 3 passed

T1.2.4: Test move()

```
function move(obj)
% Update height as the elevator moving
    % Accelerate
    if obj.accelerate_status ~= -1 && obj.moving_direction ~= 0 % Branch-Tcover1.2.4.1
        obj.Accelerate();
    end
    % Decelerate
    if obj.accelerate_status ~= 1  % Branch - Tcover1.2.4.2
        % moving down
        if obj.moving_direction==-1 && -0.000001 < obj.height-obj.floor_target &&
obj.height-obj.floor_target <= 0.1+0.000001  % Branch - Tcover1.2.4.3
            obj.Decelerate();
        end
        % moving up
        if obj.moving_direction==1 && -0.000001 < obj.floor_target-obj.height &&
obj.floor_target-obj.height <= 0.1+0.000001  % Branch - Tcover1.2.4.4
            obj.Decelerate();
        end
    end
    % Update height
    switch obj.moving_direction
        case 1  % moving upward
            if obj.height+obj.velocity <= obj.height_limit(2) % Branch - Tcover1.2.4.5
                obj.height = obj.height + obj.velocity;
            else  % Branch - Tcover1.2.4.6
                obj.height = obj.height_limit(2);
            end
        case -1 % moving downward
            if obj.height-obj.velocity >= obj.height_limit(1) % Branch - Tcover1.2.4.7
                obj.height = obj.height - obj.velocity;
            else  % Branch - Tcover1.2.4.8
                obj.height = obj.height_limit(1);
            end
    end
end
```

- Coverage Criteria: Branch coverage
- Test case

Test Case	Test Case T1.2.4.1	Test Case T1.2.4.2	Test Case T1.2.4.3	Test Case T1.2.4.4
Coverage Item	TCover1.2.4.1.1, TCover1.2.4.2.2, TCover1.2.4.5.1	TCover1.2.4.1.2, TCover1.2.4.2.1, TCover1.2.4.3.1, TCover1.2.4.4.2, TCover1.2.4.7.1	TCover1.2.4.1.2, TCover1.2.4.2.1, TCover1.2.4.3.2, TCover1.2.4.4.1, TCover1.2.4.5.2, TCover1.2.4.6.1	TCover1.2.4.1.2, TCover1.2.4.2.1, TCover1.2.4.3.1, TCover1.2.4.4.2, TCover1.2.4.7.2, TCover1.2.4.8.1
Input	—	—	—	—
State	velocity=0.1; moving_direction=1 height=2.0	velocity=0.2; moving_direction=-1; height=2.1	velocity=0.1; moving_direction=1 height=2.9	velocity=0.1; moving_direction=1 height=3.1
Expected Output	height=2.1	height=2.0	height=3.0	height=3.0

- Test coverage: 14/14 = 100%
- Test result: 4 passed

T1.2.5: Test updateFloor()

```
function updateFloor(obj)
    if obj.moving_direction == 1   % is moving up  % Branch - Tcover1.2.5.1
        obj.floor = floor(obj.height+0.000001);
    elseif obj.moving_direction == -1  % is moving down  % Branch - Tcover1.2.5.2
        obj.floor = ceil(obj.height-0.000001);
    else  % is still  % Branch - Tcover1.2.5.3
        obj.floor = floor(obj.height+0.000001);
    end
end
```

- Coverage Criteria: Branch coverage

- Test case

Test Case	Test Case T1.2.5.1	Test Case T1.2.5.2	Test Case T1.2.5.3
Coverage Item	TCover1.2.5.1.1	TCover1.2.5.1.2, TCover1.2.5.2.1	TCover1.2.5.1.2, TCover1.2.5.2.2, TCover1.2.5.3.1
Input	—	—	—
State	moving_direction=1	moving_direction=-1	moving_direction=0
Expected Output	Floor=floor(height)	Floor=ceil(height)	Floor=floor

- Test coverage: 5/5 = 100%
- Test result: 3 passed

T1.2.6: Test updateQueue()

```
function updateQueue(obj)
    % update moving_up_queue & moving_down_queue
    if obj.moving_direction==1 && isempty(obj.moving_up_queue)==0  % Branch -
Tcover1.2.6.1
        if obj.floor == obj.moving_up_queue(1) && obj.height-0.000001 <
obj.moving_up_queue(1)  % Branch - Tcover1.2.6.2
            obj.moving_up_queue(1) = [];
            % Store direction
            obj.direction_store = obj.moving_direction;
            obj.door_open = 1;
            obj.hSystemController.hFloorUI(obj.floor+1).lightOff(1,obj.floor);
        end
    elseif obj.moving_direction==-1 && isempty(obj.moving_down_queue)==0  % Branch -
Tcover1.2.6.3
        if obj.floor == obj.moving_down_queue(1) && obj.height+0.000001 >
obj.moving_down_queue(1)  % Branch - Tcover1.2.6.4
            obj.moving_down_queue(1) = [];
            % Store direction
            obj.direction_store = obj.moving_direction;
            obj.door_open = 1;
            obj.hSystemController.hFloorUI(obj.floor+1).lightOff(-1,obj.floor);
        end
    elseif obj.moving_direction==0  % Branch - Tcover1.2.6.5
        % At up start place
        if isempty(obj.moving_up_queue)==0  % Branch - Tcover1.2.6.6
            if obj.floor == obj.moving_up_queue(1)  % Branch - Tcover1.2.6.7
                obj.moving_up_queue(1) = [];
                obj.door_open = 1;
                obj.hSystemController.hFloorUI(obj.floor+1).lightOff(1,obj.floor);
                return
            end
        end
        % At down start place
        if isempty(obj.moving_down_queue)==0  % Branch - Tcover1.2.6.8
            if obj.floor == obj.moving_down_queue(1)  % Branch - Tcover1.2.6.9
                obj.moving_down_queue(1) = [];
                obj.door_open = 1;
                obj.hSystemController.hFloorUI(obj.floor+1).lightOff(-1,obj.floor);
                return
            end
        end
    end
end
```

- Coverage Criteria: Branch coverage

- Test case

Test Case	Test Case T1.2.6.1	Test Case T1.2.6.2	Test Case T1.2.6.3	Test Case T1.2.6.4
Coverage Item	TCover1.2.6.1.1, TCover1.2.6.2.1	TCover1.2.6.1.1, TCover1.2.6.2.2	TCover1.2.6.1.2, TCover1.2.6.3.1, TCover1.2.6.4.1	TCover1.2.6.1.2, TCover1.2.6.3.1, TCover1.2.6.4.2
Input	—	—	—	—
State	moving_direction=1 moving_up_queue = [2]; height=2.0	moving_direction=1 moving_up_queue = [2]; height=1.7	moving_direction = -1; moving_down_queue = [2]; height=2.0	moving_direction = -1; moving_down_queue = [2]; height=2.7
Expected Output	Door_open=1	Door_open=0	Door_open=1	Door_open=0
Test Case	Test Case T1.2.6.5	Test Case T1.2.6.6	Test Case T1.2.6.7	Test Case T1.2.6.8
Coverage Item	TCover1.2.6.1.2, TCover1.2.6.3.2, TCover1.2.6.5.1, TCover1.2.6.6.1, TCover1.2.6.7.1	TCover1.2.6.1.2, TCover1.2.6.3.2, TCover1.2.6.5.1, TCover1.2.6.6.1, TCover1.2.6.7.2, TCover1.2.6.8.1, TCover1.2.6.9.1	TCover1.2.6.1.2, TCover1.2.6.3.2, TCover1.2.6.5.1, TCover1.2.6.6.2, TCover1.2.6.8.1, TCover1.2.6.9.2	TCover1.2.6.1.2, TCover1.2.6.3.2, TCover1.2.6.5.1, TCover1.2.6.6.2, TCover1.2.6.8.2
Input	—	—	—	—
State	moving_direction=0 moving_up_queue = [2]; height=2.0	moving_direction=0 moving_down_queue = [2]; height=2.0	moving_direction=0 moving_down_queue = [2]; height=2.3	moving_direction=0 moving_up_queue = []; moving_down_queue = []; height=2.0
Expected Output	Door_open=1	Door_open=1	Door_open=0	Door_open=0
Test Case	Test Case T1.2.6.9			
Coverage Item	TCover1.2.6.1.2, TCover1.2.6.3.2, TCover1.2.6.5.2			
Input	—			
State	moving_direction=1 moving_up_queue = []; moving_down_queue = []; height=2.0			
Expected Output	Door_open=0			

- Test coverage: 18/18 = 100%
- Test result: 9 passed

T1.2.7: Test updateTargetFloor()

```
function updateTargetFloor(obj)
    % Moving up
    if obj.moving_direction == 1  % Branch - Tcover1.2.7.1
```

```
        if isempty(obj.moving_up_queue)==0  % Branch - Tcover1.2.7.2
            obj.floor_target = obj.moving_up_queue(1);
        elseif isempty(obj.moving_down_queue)==0  % Branch - Tcover1.2.7.3
            obj.floor_target = obj.moving_down_queue(1);
        end
    elseif obj.moving_direction == -1  % Branch - Tcover1.2.7.4
        if isempty(obj.moving_down_queue)==0  % Branch - Tcover1.2.7.5
            obj.floor_target = obj.moving_down_queue(1);
        elseif isempty(obj.moving_up_queue)==0  % Branch - Tcover1.2.7.6
            obj.floor_target = obj.moving_up_queue(1);
        end
    end
end
```

- Coverage Criteria: Branch coverage
- Test case

Test Case	Test Case T1.2.7.1	Test Case T1.2.7.2	Test Case T1.2.7.3	Test Case T1.2.7.4
Coverage Item	TCover1.2.7.1.1, TCover1.2.7.2.1	TCover1.2.7.1.1, TCover1.2.7.2.2, TCover1.2.7.3.1	TCover1.2.7.1.1, TCover1.2.7.2.2, TCover1.2.7.3.2	TCover1.2.7.1.2, TCover1.2.7.4.1, TCover1.2.7.5.1
Input	—	—	—	—
State	moving_direction=1 moving_up_queue = [2]; moving_down_queue = [];	moving_direction=1 moving_up_queue = []; moving_down_queue = [2];	moving_direction=1 moving_up_queue = []; moving_down_queue = [];	moving_direction=-1; moving_up_queue = []; moving_down_queue = [2];
Expected Output	Floor_target = 2;	Floor_target = 2;	—	Floor_target = 2;
Test Case	Test Case T1.2.7.5	Test Case T1.2.7.6	Test Case T1.2.7.7	
Coverage Item	TCover1.2.7.1.2, TCover1.2.7.4.1, TCover1.2.7.5.2, TCover1.2.7.6.1	TCover1.2.7.1.2, TCover1.2.7.4.1, TCover1.2.7.5.2, TCover1.2.7.6.2	TCover1.2.7.1.2, TCover1.2.7.4.2	
Input	—	—	—	—
State	moving_direction=-1; moving_up_queue = [2]; moving_down_queue = [];	moving_direction=-1; moving_up_queue = []; moving_down_queue = [];	moving_direction=0 moving_up_queue = []; moving_down_queue = [];	
Expected Output	Floor_target = 2;	—	—	

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

T1.2.8: Test addSchedule()

```
function addSchedule(obj, target_floor, direction_call)
% Add target floor to the queue
    if direction_call == "up"  % Branch - Tcover1.2.8.1
        obj.moving_up_queue = [obj.moving_up_queue target_floor];
        obj.moving_up_queue = sort(obj.moving_up_queue, 'ascend');
    elseif direction_call == "down"  % Branch - Tcover1.2.8.2
        obj.moving_down_queue = [obj.moving_down_queue target_floor];
        obj.moving_down_queue = sort(obj.moving_down_queue, 'descend');
    end
end
```

- Coverage Criteria: Branch coverage
- Test case

Test Case	Test Case T1.2.8.1	Test Case T1.2.8.2
Coverage Item	TCover1.2.8.1.1	TCover1.2.8.1.2, TCover1.2.8.2.1
Input	[2, ‘up’]	[2,’down’]
State	Moving_up_queue=[1]	Moving_down_queue=[1]
Expected Output	Moving_up_queue=[1,2]	Moving_down_queue=[2,1]

- Test coverage: 3/3 = 100%
- Test result: 2 passed

T1.3: CallUI Test

T1.3.1: Test floorCall()

```
function floorCall(app, target_floor)
% Add the call target floor to the Elevator moving queue
    if target_floor <= app.hElevator.height + 0.1  % Branch - Tcover1.3.1.1
        app.hElevator.moving_down_queue = [app.hElevator.moving_down_queue
target_floor];
        app.hElevator.moving_down_queue = sort(app.hElevator.moving_down_queue,
'descend');
    elseif target_floor > app.hElevator.height - 0.1  % Branch - Tcover1.3.1.2
        app.hElevator.moving_up_queue = [app.hElevator.moving_up_queue target_floor];
        app.hElevator.moving_up_queue = sort(app.hElevator.moving_up_queue, 'ascend');
    end
end
```

- Coverage Criteria: Branch coverage
- Test case

Test Case	Test Case T1.3.1.1	Test Case T1.3.1.2
Coverage Item	TCover1.3.1.1.1	TCover1.3.1.1.2, TCover1.3.1.2.1
Input	1	3
State	Moving_down_queue=[2], Height = 2.3	Moving_up_queue=[1], Height = 2.0
Expected Output	Moving_down_queue=[2,1]	Moving_up_queue=[1,3]

- Test coverage: 3/3 = 100%
- Test result: 2 passed

T1.3.2: Test updateDisplay()

```
function updateDisplay(app)
    % Moving direction
    if app.hElevator.moving_direction == 1  % Branch - Tcover1.3.2.1
        moving_direction = '▲';
    elseif app.hElevator.moving_direction == -1  % Branch - Tcover1.3.2.2
        moving_direction = '▼';
    else  % Branch - Tcover1.3.2.3
        moving_direction = '';
    end

    % Floor display
    if app.hElevator.floor == 0  % Branch - Tcover1.3.2.4
        app.floorDisplay.Text = [moving_direction, 'B'];
    else  % Branch - Tcover1.3.2.5
        app.floorDisplay.Text = [moving_direction, num2str(app.hElevator.floor)];
    end
    % Overweight
    if app.hElevator.over_weight == 1  % Branch - Tcover1.3.2.6
        app.OverweightLamp.Color = [1,0,0];
    else  % Branch - Tcover1.3.2.7
        app.OverweightLamp.Color = [0.8,0.8,0.8];
    end
    % Emergency
    if app.hElevator.emergency == 1  % Branch - Tcover1.3.2.8
        app.EmergencyLamp.Color = [1,0,0];
    else  % Branch - Tcover1.3.2.9
        app.EmergencyLamp.Color = [0.8,0.8,0.8];
    end
    % Maintain
    if app.hElevator.maintain == 1  % Branch - Tcover1.3.2.10
        app.floorDisplay.Text = "M";
    end
    % Button display
    if app.hElevator.moving_direction == 0  % Branch - Tcover1.3.2.11
        if abs(app.hElevator.floor - app.hElevator.height) < 0.000001  % Branch -
Tcover1.3.2.12
            if app.hElevator.floor == 0    % Branch - Tcover1.3.2.13
                app.FB_0.Visible = 1;
                app.FB_1.Visible = 0;
            else    % Branch - Tcover1.3.2.14
                floor = num2str(app.hElevator.floor);
                eval(['app.F',floor,'_0.Visible = 1;']);
                eval(['app.F',floor,'_1.Visible = 0;']);
            end
        end
    end
end
```

- Coverage Criteria: Branch coverage

- Test case

Test Case	Test Case T1.3.2.1	Test Case T1.3.2.2	Test Case T1.3.2.3	Test Case T1.3.2.4
Coverage Item	TCover1.3.2.1.1, TCover1.3.2.4.1, TCover1.3.2.6.2, TCover1.3.2.7.1, TCover1.3.2.8.1, TCover1.3.2.10.1, TCover1.3.2.11.2	TCover1.3.2.1.2, TCover1.3.2.2.1, TCover1.3.2.4.2, TCover1.3.2.5.1, TCover1.3.2.6.1, TCover1.3.2.8.2, TCover1.3.2.9.1, TCover1.3.2.10.2, TCover1.3.2.11.2	TCover1.3.2.1.2, TCover1.3.2.2.2, TCover1.3.2.3.1, TCover1.3.2.4.1, TCover1.3.2.6.2, TCover1.3.2.7.1, TCover1.3.2.8.2, TCover1.3.2.9.1, TCover1.3.2.10.2, TCover1.3.2.11.1, TCover1.3.2.12.2	TCover1.3.2.1.2, TCover1.3.2.2.2, TCover1.3.2.3.1, TCover1.3.2.4.1, TCover1.3.2.6.2, TCover1.3.2.7.1, TCover1.3.2.8.2, TCover1.3.2.9.1, TCover1.3.2.10.2, TCover1.3.2.11.1, TCover1.3.2.12.1, TCover1.3.2.13.1
Input	—	—	—	—
State	moving_direction=1 floor = 0; over_weight=0; emergency=1; maintenance=1;	moving_direction = -1; floor = 1; over_weight=1; emergency=0; maintenance=0;	moving_direction=0 floor = 0; over_weight=0; emergency=0; maintenance=0; height=0.5;	moving_direction=0 floor = 0; over_weight=0; emergency=0; maintenance=0; height = 0.0;
Expected Output	—	—	—	—
Test Case	Test Case T1.3.2.5			
Coverage Item	TCover1.3.2.1.2, TCover1.3.2.2.2, TCover1.3.2.3.1, TCover1.3.2.4.1, TCover1.3.2.6.2, TCover1.3.2.7.1, TCover1.3.2.8.2, TCover1.3.2.9.1, TCover1.3.2.10.2, TCover1.3.2.11.1, TCover1.3.2.12.1, TCover1.3.2.13.2, TCover1.3.2.14.1			
Input	—			
State	moving_direction=0 floor = 1; over_weight=0; emergency=0; maintenance=0; height = 1.0;			
Expected Output	—			

- Test coverage: 23/23 = 100%
- Test result: 5 passed

T1.3.3: Test doorOpen()

```
function doorOpen(app)
    % If need reset
    if app.hElevator.timer_door >= 8  % Branch - Tcover1.3.3.1
        app.hElevator.door_open = 0;
        app.hElevator.timer_door = 0;
        app.hElevator.moving_direction = app.hElevator.direction_store;
        app.hElevator.direction_store = 0;
        app.hElevator.movingDirector();   % get direction
    end
    if app.hElevator.door_open == 1  % Branch - Tcover1.3.3.2
        % Store moving_direction
        if app.hElevator.moving_direction ~= 0 && app.hElevator.direction_store==0  %
Branch - Tcover1.3.3.3
            app.hElevator.moving_direction = 0;
        end
        % Opening
        if app.hElevator.timer_door < 1  % Branch - Tcover1.3.3.4
            app.hElevator.door_status = "opening";
            app.open_1.Visible = 1;
            app.open_0.Visible = 0;
            app.close_0.Visible = 1;
            app.close_1.Visible = 0;
            app.DoorOpenLamp.Color = [1,0,0];
            app.DoorCloseLamp.Color = [0.8,0.8,0.8];
            app.left_door.Position = [46,33,99,254];
            app.right_door.Position = [265,33,99,254];
        % Is Open
        elseif 1 <= app.hElevator.timer_door && app.hElevator.timer_door < 6  % Branch
- Tcover1.3.3.5
            app.hElevator.door_status = "open";
            app.open_1.Visible = 0;
            app.open_0.Visible = 1;
            app.close_0.Visible = 1;
            app.close_1.Visible = 0;
            app.DoorOpenLamp.Color = [0,1,0];
            app.DoorCloseLamp.Color = [0.8,0.8,0.8];
            app.left_door.Position = [46,33,29,254];
            app.right_door.Position = [335,33,29,254];
        % Closing
        elseif 6 <= app.hElevator.timer_door && app.hElevator.timer_door < 7  % Branch
- Tcover1.3.3.6
            app.hElevator.door_status = "closing";
            app.open_1.Visible = 0;
            app.open_0.Visible = 1;
            app.close_1.Visible = 1;
            app.close_0.Visible = 0;
            app.DoorOpenLamp.Color = [0.8,0.8,0.8];
            app.DoorCloseLamp.Color = [1,0,0];
            app.left_door.Position = [46,33,99,254];
            app.right_door.Position = [265,33,99,254];
        % Close
        elseif app.hElevator.timer_door >= 7  % Branch - Tcover1.3.3.7
            app.hElevator.door_status = "close";
            app.open_1.Visible = 0;
            app.open_0.Visible = 1;
            app.close_0.Visible = 1;
            app.close_1.Visible = 0;
            app.DoorCloseLamp.Color = [0,1,0];
            app.DoorOpenLamp.Color = [0.8,0.8,0.8];
            app.left_door.Position = [46,33,169,254];
            app.right_door.Position = [195,33,169,254];
        end
        % Update timer
        app.hElevator.timer_door = app.hElevator.timer_door + 1;
    end
end
```

- Coverage Criteria: Branch coverage
- Test case

Test Case	Test Case T1.3.3.1	Test Case T1.3.3.2	Test Case T1.3.3.3	Test Case T1.3.3.4
Coverage Item	TCover1.3.3.1.2, TCover1.3.3.2.1, TCover1.3.3.3.1, TCover1.3.3.4.1, TCover1.3.3.5.2, TCover1.3.3.6.2, TCover1.3.3.7.2	TCover1.3.3.1.2, TCover1.3.3.2.1, TCover1.3.3.3.2, TCover1.3.3.4.2, TCover1.3.3.5.1, TCover1.3.3.6.2, TCover1.3.3.7.2	TCover1.3.3.1.2, TCover1.3.3.2.1, TCover1.3.3.3.2, TCover1.3.3.4.2, TCover1.3.3.5.2, TCover1.3.3.6.1, TCover1.3.3.7.2	TCover1.3.3.1.2, TCover1.3.3.2.1, TCover1.3.3.3.2, TCover1.3.3.4.2, TCover1.3.3.5.2, TCover1.3.3.6.2, TCover1.3.3.7.1
Input	—	—	—	—
State	Moving_direction=1 Timer_door=0	Moving_direction=0 Timer_door=3	Moving_direction=0 Timer_door=6	Moving_direction=0 Timer_door=7
Expected Output	Moving_direction=0 Timer_door=1	Moving_direction=0 Timer_door=4	Moving_direction=0 Timer_door=7	Moving_direction=0 Timer_door=8
Test Case	Test Case T1.3.3.5	Test Case T1.3.3.6		
Coverage Item	TCover1.3.3.1.1, TCover1.3.3.2.1, TCover1.3.3.3.2, TCover1.3.3.4.2, TCover1.3.3.5.2, TCover1.3.3.6.2, TCover1.3.3.7.2	TCover1.3.3.1.2, TCover1.3.3.2.2		
Input	—	—		
State	Moving_direction=0	Door_open=0;		

	Timer_door=8	Timer_door=0		
Expected Output	Moving_direction=1 Timer_door=0; Door_open=0	Timer_door=0; Door_open=0		

- Test coverage: 14/14 = 100%
- Test result: 6 passed

T1.4: FloorUI Test

T1.4.1: Test updateDisplay1()

Note: Here we only shows the update process of elevator #1. For #2, the process is completely the same, so it is omitted here.

```
function updateDispaly1(app)
    % Elevator 1
    if app.hElevators(1).moving_direction == 1  % Branch - Tcover1.4.1.1
        moving_direction = '▲';
    elseif app.hElevators(1).moving_direction == -1  % Branch - Tcover1.4.1.2
        moving_direction = '▼';
    else  % Branch - Tcover1.4.1.3
        moving_direction = '';
    end
    if app.hElevators(1).maintain == 1  % Branch - Tcover1.4.1.4
        app.F1_display_1.Text = "M";
    else  % Branch - Tcover1.4.1.5
        if app.hElevators(1).floor == 0  % Branch - Tcover1.4.1.6
            app.F1_display_1.Text = [moving_direction, 'B'];
        else  % Branch - Tcover1.4.1.7
            app.F1_display_1.Text = [moving_direction,
num2str(app.hElevators(1).floor)];
        end
     end
end
```

- Coverage Criteria: Branch coverage
- Test case

Test Case	Test Case T1.4.1.1	Test Case T1.4.1.2	Test Case T1.4.1.3
Coverage Item	TCover1.4.1.1.1, TCover1.4.1.4.1	TCover1.4.1.1.2, TCover1.4.1.2.1, TCover1.4.1.4.2, TCover1.4.1.5.1, TCover1.4.1.6.1	TCover1.4.1.1.2, TCover1.4.1.2.2, TCover1.4.1.3.1, TCover1.4.1.4.2, TCover1.4.1.5.1, TCover1.4.1.6.2, TCover1.4.1.7.1
Input	—	—	—
State	Moving_direction=1; Maintenance=1;	Moving_direction=-1; Maintenance=0;	Moving_direction=0; Maintenance=0;

	Floor=2;	Floor=0;	Floor=2;
Expected Output	—	—	—

- Test coverage: 11/11 = 100%
- Test result: 3 passed

T1.4.2: Test lightOff()

```
function lightOff(app, direction, floor)
    % Set the button status to 0
    if floor == app.floor  % Branch - Tcover1.4.2.1
        if direction == 1 && app.floor < 3   % up  % Branch - Tcover1.4.2.2
            app.F1_up_0.Visible = 1;
            app.F1_up_1.Visible = 0;
        elseif direction == -1 && app.floor > 0  % down  % Branch - Tcover1.4.2.3
            app.F1_down_0.Visible = 1;
            app.F1_down_1.Visible = 0;
        end
    end
end
```

- Coverage Criteria: Branch coverage
- Test case

Test Case	Test Case T1.4.2.1	Test Case T1.4.2.2	Test Case T1.4.2.3	Test Case T1.4.2.4
Coverage Item	TCover1.4.2.1.1, TCover1.4.2.2.1	TCover1.4.2.1.1, TCover1.4.2.2.2, TCover1.4.2.3.1	TCover1.4.2.1.1, TCover1.4.2.2.2, TCover1.4.2.3.2	TCover1.4.2.1.2
Input	Floor=2; Direction=1;	Floor=2; Direction=-1;	Floor=3; Direction=1;	Floor=0; Direction=1;
State	App.floor=2	App.floor=2	App.floor=3	App.floor=2
Expected Output	—	—	—	—

- Test coverage: 6/6 = 100%
- Test result: 4 passed

T1.5: ActivityMonitor Test

T1.5.1: Test updateSpeed()

```
function updateSpeed(app)
    app.SpeedSlider.Value = app.hElevator.velocity * 
app.hElevator.moving_direction;  % Statement - Tcover1.5.1.1
end
```

- Coverage Criteria: Branch coverage
- Test case

Test Case	Test Case T1.5.1.1
Coverage Item	TCover1.5.1.1.1
Input	—
State	app.hElevator.velocity = 0.2; app.hElevator.moving_direction = -1;
Expected Output	SpeedSlider.Value = velocity * direction;

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

T1.5.2: Test doorOpen()

```
function doorOpen(app)
    % If need reset
    if app.hElevator.door_open == 1  % Branch - Tcover1.5.2.1
        % Opening
        if app.hElevator.timer_door-1 < 1  % Branch - Tcover1.5.2.2
            app.DoorOpenLamp.Color = [1,0,0];
            app.DoorCloseLamp.Color = [0.8,0.8,0.8];
        % Is Open
        elseif 1 <= app.hElevator.timer_door-1 && app.hElevator.timer_door-1 < 6  %
Branch - Tcover1.5.2.3
            app.DoorOpenLamp.Color = [0,1,0];
        % Closing
        elseif 6 <= app.hElevator.timer_door-1 && app.hElevator.timer_door-1 < 7  %
Branch - Tcover1.5.2.4
            app.DoorOpenLamp.Color = [0.8,0.8,0.8];
            app.DoorCloseLamp.Color = [1,0,0];
        % Close
        elseif app.hElevator.timer_door-1 >= 7  % Branch - Tcover1.5.2.5
            app.DoorCloseLamp.Color = [0,1,0];
        end
    end
end
```

- Coverage Criteria: Branch coverage
- Test case

Test Case	Test Case T1.5.2.1	Test Case T1.5.2.2	Test Case T1.5.2.3	Test Case T1.5.2.4
Coverage Item	TCover1.5.2.1.1, TCover1.5.2.2.1	TCover1.5.2.1.1, TCover1.5.2.2.2, TCover1.5.2.3.1	TCover1.5.2.1.1, TCover1.5.2.2.2, TCover1.5.2.3.2, TCover1.5.2.4.1	TCover1.5.2.1.1, TCover1.5.2.2.2, TCover1.5.2.3.2, TCover1.5.2.4.2, TCover1.5.2.5.1
Input	—	—	—	—
State	Door_open=1; Timer_door=1;	Door_open=1; Timer_door=5;	Door_open=1; Timer_door=7;	Door_open=1; Timer_door=8;
Expected Output	app.DoorOpenLamp .Color = [1,0,0];	app.DoorOpenLamp .Color = [0,1,0];	app.DoorCloseLamp .Color = [1,0,0];	app.DoorCloseLamp .Color = [0,1,0];

	app.DoorCloseLamp .Color = [0.8,0.8,0.8];		app.DoorOpenLamp .Color =[0.8,0.8,0.8];	
Test Case	Test Case T1.5.2.5			
Coverage Item	TCover1.5.2.1.2			
Input	—			
State	Door_open=0; Timer_door=0;			
Expected Output	None			

- Test coverage: 9/9 = 100%
- Test result: 5 passed

T2: Integration Test

T2.1: SystemController + 4FloorUI Integration

T2.1.1: Test Floor Call with one Elevator

```
function OutCall_1(tc)
    % T2.1.1: Floor Call with one Elevator
    tc.press(tc.hFloorUI(4).F1_down_0);
    pause(0.5);
    tc.press(tc.hFloorUI(3).F1_down_0);
    pause(0.5);
    tc.press(tc.hFloorUI(3).F1_up_0);
end
```

- Test case

Test Case	Test Case T2.1.1.1
Coverage Item	TCover1.2.1, TCover1.2.2, TCover1.2.3, TCover1.2.4, TCover1.2.5, TCover1.2.6, TCover1.2.7, TCover1.2.8, TCover1.4.1, TCover1.4.2
Input	Press (F3, down), (F2, down), (F2, up) respectively
State	Elevator is waiting at F1
Expected Output	Elevator goes to F2, F3, F2 and stopeed respectively

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

T2.1.2: Test Floor Call with two Elevator

```
function OutCall_2(tc)
    % T2.1.2: Floor Call with two Elevators
    tc.press(tc.hFloorUI(4).F1_down_0);
    pause(0.5);
    tc.press(tc.hFloorUI(3).F1_down_0);
    pause(0.5);
    tc.press(tc.hFloorUI(3).F1_up_0);
end
```

- Test case

Test Case	Test Case T2.1.2.1
Coverage Item	TCover1.1.1, TCover1.1.2, TCover1.2.1, TCover1.2.2, TCover1.2.3, TCover1.2.4, TCover1.2.5, TCover1.2.6, TCover1.2.7, TCover1.2.8, TCover1.4.1, TCover1.4.2
Input	Press (F3, down), (F2, down), (F2, up) respectively
State	Elevators are waiting at F1
Expected Output	Elevator1 goes to F3, F2 and stopped respectively; Elevator2 goes to F2 and stopped.

- Test coverage: 12/12 = 100%
- Test result: 1 passed

T2.2: SystemController + 2CarUI Integration

T2.2.1: Test Door Control

```
function InCall_1(tc)
    % T2.1.2: Floor Selection two Elevators

    % Door Open
    tc.press(tc.hElevators(1).hCarUI.F2_0);
    pause(1);
    tc.press(tc.hElevators(1).hCarUI.open_0);
    pause(0.5);
    tc.press(tc.hElevators(1).hCarUI.open_0);
    pause(0.5);
    tc.press(tc.hElevators(1).hCarUI.open_0);
    pause(0.5);
    tc.press(tc.hElevators(1).hCarUI.open_0);
    pause(0.5);
    tc.press(tc.hElevators(1).hCarUI.open_0);
    pause(0.5);
    tc.press(tc.hElevators(1).hCarUI.open_0);
    pause(0.5);
    tc.press(tc.hElevators(1).hCarUI.open_0);
    pause(1);

    % Door Close
    tc.press(tc.hElevators(1).hCarUI.F3_0);
    pause(1);
    tc.press(tc.hElevators(1).hCarUI.close_0);
    pause(0.5);
    tc.press(tc.hElevators(1).hCarUI.close_0);
    pause(0.5);
    tc.press(tc.hElevators(1).hCarUI.close_0);
    pause(0.5);
    tc.press(tc.hElevators(1).hCarUI.close_0);
    pause(0.5);
    tc.press(tc.hElevators(1).hCarUI.close_0);
    pause(0.5);
    tc.press(tc.hElevators(1).hCarUI.close_0);
    pause(0.5);
    tc.press(tc.hElevators(1).hCarUI.close_0);
    pause(1);

end
```

- Test case

Test Case	Test Case T2.2.1.1
Coverage Item	TCover1.2.1, TCover1.2.4, TCover1.2.8, TCover1.3.1, TCover1.3.3

Input	Press (F2), (door_open), (door_open), (door_open), (door_open), (door_open), (door_open), (door_open);
	Then press (F3), (door_close), (door_close), (door_close), (door_close), (door_close), (door_close), (door_close);

State	Elevator is waiting at F1
Expected Output	The door only opens when the elevator is stopping at floor 2 and 3

- Test coverage: 5/5 = 100%
- Test result: 1 passed

T2.2.2: Test Floor Selection

```
function InCall_1(tc)
    % T2.1.2: Floor Selection two Elevators

    % Elevator 1
    tc.press(tc.hElevators(1).hCarUI.F3_0);
    pause(1);
    tc.press(tc.hElevators(1).hCarUI.F2_0);
    pause(1);
    tc.press(tc.hElevators(1).hCarUI.F1_0);
    pause(1);

    % Elevator 2
    tc.press(tc.hElevators(2).hCarUI.F2_0);
    pause(1);
    tc.press(tc.hElevators(2).hCarUI.F1_0);
    pause(1);
    tc.press(tc.hElevators(2).hCarUI.FB_0);
    pause(1);

end
```

- Test case

Test Case	Test Case T2.2.2.1
Coverage Item	TCover1.2.1, TCover1.2.2, TCover1.2.3, TCover1.2.4, TCover1.2.5, TCover1.2.6, TCover1.2.7, TCover1.2.8, TCover1.3.1, TCover1.3.2, TCover1.3.3
Input	Elevator 1: Press (F3), (F2), (F1) respectively; Elevator 2: Press (F2), (F1), (FB) respectively.

State	Elevators are waiting at F1
Expected Output	Elevator1 goes to F3, F2, F1 and stopped respectively; Elevator2 goes to F2, F1, FB and stopped respectively.

- Test coverage: 11/11 = 100%
- Test result: 1 passed

T2.3: SystemController + 4FloorUI + 2CarUI + Monitor Integration

T2.3.1: Demo Test

```
function DemoTest_1(tc)
    % T2.3.1: Demo test

    % Floor Calls
    tc.press(tc.hFloorUI(4).F1_down_0);
    pause(1);

    tc.press(tc.hFloorUI(3).F1_up_0);
    pause(1);
    tc.press(tc.hFloorUI(3).F1_down_0);
    pause(1);

    tc.press(tc.hFloorUI(1).F1_up_0);
    pause(1);

    % Elevator 1
    tc.press(tc.hElevators(1).hCarUI.F3_0);
    pause(1);
    tc.press(tc.hElevators(1).hCarUI.F2_0);
    pause(1);
    tc.press(tc.hElevators(1).hCarUI.F1_0);
    pause(1);

    % Elevator 2
    tc.press(tc.hElevators(2).hCarUI.F2_0);
    pause(1);
    tc.press(tc.hElevators(2).hCarUI.F1_0);
    pause(1);
    tc.press(tc.hElevators(2).hCarUI.FB_0);
    pause(1);

end
```

- Test case

Test Case	Test Case T2.2.2.1
Coverage Item	TCover1.1.1, TCover1.1.2, TCover1.2.1, TCover1.2.2, TCover1.2.3, TCover1.2.4, TCover1.2.5, TCover1.2.6, TCover1.2.7, TCover1.2.8, TCover1.3.1, TCover1.3.2, TCover1.3.3, TCover1.4.1, TCover1.4.2, TCover1.5.1, TCover1.5.2

Input	Press (F3, down), (F2, up), (F2, down), (FB, up) respectively;
	Then, Elevator 1: Press (F3), (F2), (F1) respectively;

	Then, Elevator 2: Press (F2), (F1), (FB) respectively.
State	Elevators are waiting at F1
Expected Output	Elevator1 goes to F2, F3, F1, F2 and stopped respectively; Elevator2 goes to F2, F1, FB and stopped respectively.

- Test coverage: 17/17 = 100%
- Test result: 1 passed

T3: Functional Test

T3.1: Use Case 'Call one Elevator'

- Test case

Test Case	Test Case T3.1.1
State	Elevator1 stopped at F1
Operation	1. Press 'Up' button on F2; 2. Wait.
Expected Behavior	1. 'Up' button lightened, Elevator1 comes up; 2. Elevator1 arrived, 'Up' button lights off; 3. Elevator1 door opened, then closed;

- Test result: 1 passed

T3.2: Use Case 'Control Door Status'

- Test case

Test Case	Test Case T3.2.1	Test Case T3.2.2
State	Elevator1 stopped at F1, door close	Elevator1 stopped at F1, door open
Operation	1. Press 'Door Open' button on the panel; 2. Wait.	1. Press 'Door Close' button on the panel; 2. Wait.
Expected Behavior	1. 'Door Open' button lightened, Elevator1 door is opening; 2. Elevator1 door opened, 'Door Open' button lights off;	1. 'Door Close' button lightened, Elevator1 door is closing; 2. Elevator1 door closed, 'Door Close' button lights off;

- Test result: 2 passed

T3.3: Use Case 'Go to Target Floor'

T3.3.1: Single Floor

- Test case

Test Case	Test Case T3.3.1	Test Case T3.3.2
State	Elevator1 stopped at F1	Elevator2 stopped at F1
Operation	1. Press '2' button on the panel; 2. Wait.	1. Press 'B' button on the panel; 2. Wait.
Expected Behavior	1. '2' button lightened, Elevator1 went up;	1. 'B' button lightened, Elevator2 went down;

	2. Elevator1 arrived at F2; 3. Elevator1 door opened; 4. Elevator1 door closed;	2. Elevator2 arrived at FB; 3. Elevator2 door opened; 4. Elevator2 door closed;

- Test result: 2 passed

T3.3.2: Multiple Floors

- Test case

Test Case	Test Case T3.3.1	Test Case T3.3.2
State	Elevator1 stopped at F1	Elevator2 stopped at F1
Operation	1. Press ‘2’ button on the panel; 2. Press ‘3’ button on the panel; 3. Press ‘1’ button on the panel; 4. Wait.	1. Press ‘2’ button on the panel; 2. Press ‘B’ button on the panel; 3. Press ‘3’ button on the panel; 4. Press ‘1’ button on the panel; 5. Wait.
Expected Behavior	1. ‘2’ button lightened, Elevator1 went up; 2. ‘3’ button lightened; 3. ‘1’ button lightened; 4. Elevator1 arrived at F2; 5. Elevator1 door opened; 6. Elevator1 door closed; 7. Elevator1 arrived at F3; 8. Elevator1 door opened; 9. Elevator1 door closed; 10. Elevator1 went down; 11. Elevator1 arrived at F1; 12. Elevator1 door opened; 13. Elevator1 door closed;	1. ‘2’ button lightened, Elevator1 went up; 2. ‘B’ button lightened; 3. ‘3’ button lightened; 4. ‘1’ button lightened; 5. Elevator2 arrived at F2; 6. Elevator2 door opened; 7. Elevator2 door closed; 8. Elevator2 arrived at F3; 9. Elevator2 door opened; 10. Elevator2 door closed; 11. Elevator2 went down; 12. Elevator2 arrived at F1; 13. Elevator2 door opened; 14. Elevator2 door closed; 15. Elevator2 arrived at FB; 16. Elevator2 door opened; 17. Elevator2 door closed;

- Test result: 2 passed

T3.4: Use Case ‘Multi-calls’

- Test case

Test Case	Test Case T3.4.1	Test Case T3.4.2
State	Elevator1 stopped at F1, Elevator2 stopped at F1	Elevator1 stopped at F1, Elevator2 stopped at F1
Operation	1. Press ‘Up’ button on F2; 2. Press ‘Down’ button on F2; 3. Wait.	1. Press ‘Up’ button on FB; 2. Press ‘Down’ button on F3; 3. Wait.
Expected Behavior	1. ‘Up’ button lightened, Elevator1 comes up; 2. ‘Down’ button lightened, Elevator2 comes up;	1. ‘Up’ button lightened, Elevator2 comes down; 2. ‘Down’ button lightened, Elevator1 comes up;

	3. Elevator1 arrived, 'Up' button lights off; 4. Elevator2 arrived, 'Down' button lights off; 5. Elevator1 door opened, then closed; 6. Elevator2 door opened, then closed;	3. Elevator2 arrived, 'Up' button lights off; 4. Elevator2 door opened, then closed; 5. Elevator1 arrived, 'Down' button lights off; 6. Elevator1 door opened, then closed;

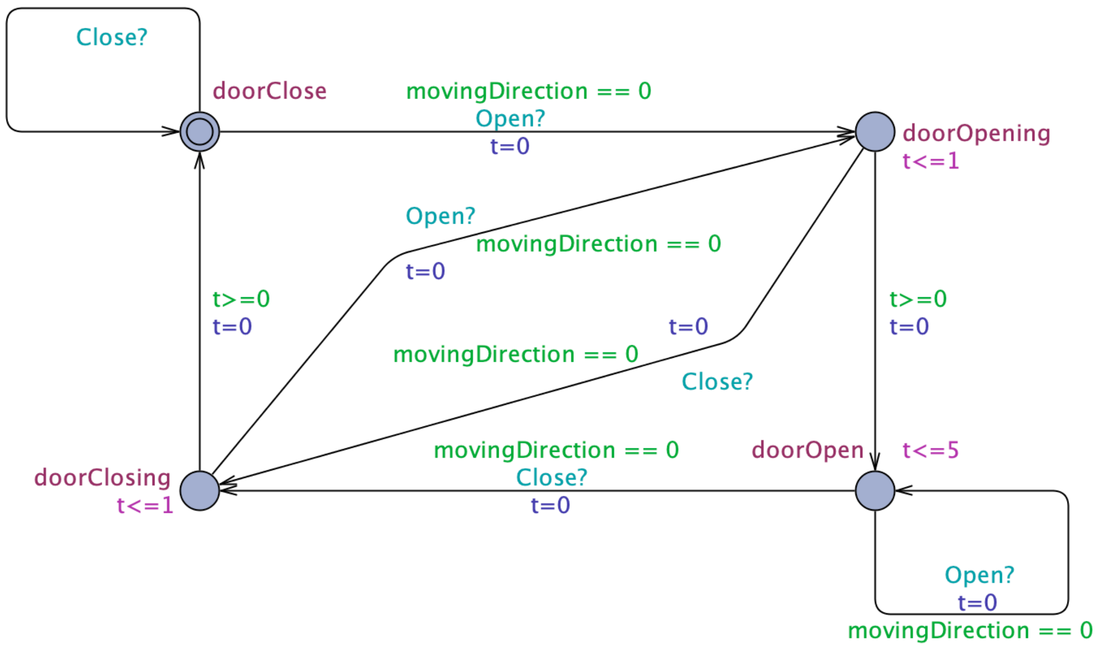
- Test result: 2 passed

Model Checking

M1: Door

M1.1: Simulator

The door starts at 'doorClose' status. When receiving the 'Open' message, the door will transfer to 'doorOpening' status, then turns to 'doorOpen'. When receiving the 'Close' message, the door will transfer to 'doorClosing' status, then turns to the initial status 'doorClose'.
If message 'Open' is received and the door is closed or closing, the door status will transfer to opening; If message 'Close' is received and the door is open or opening, the door status will transfer to closing.



M1.2: Verifier

The following properties are checked.

M1.2.1

Property	A[] not deadlock

Description	The system will not crash or dead locked.
Result	Passed

M1.2.2

Property	E<> door.doorOpen
Description	The door will be opened by the system at some moment.
Result	Passed

M1.2.3

Property	E<> door.doorClose
Description	The door will be closed by the system at some moment.
Result	Passed

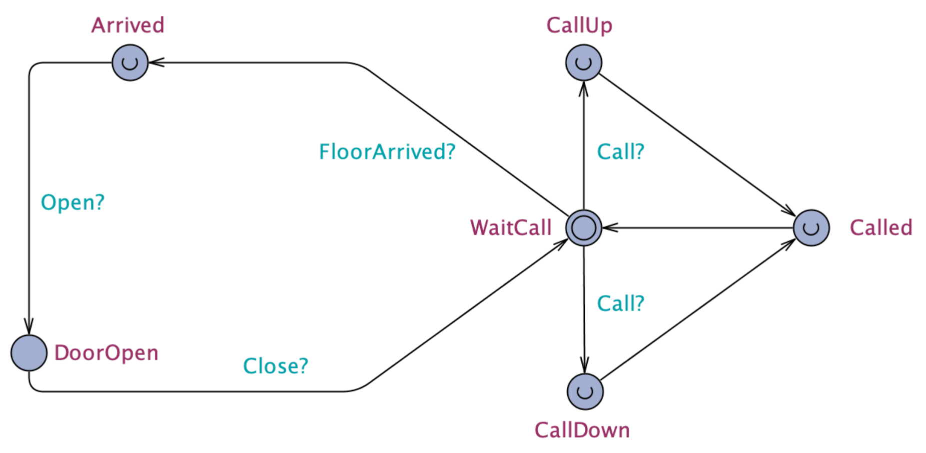
M1.2.4

Property	A[] door.t <= 5
Description	The opening time for the door is no longer than 5 seconds.
Result	Passed

M2: FloorUI

M2.1: Simulator

Users can call elevators through FloorUI, including CallUp and CallDown. Moreover, the door status is displayed in the FloorUI.
In this project, 4 FloorUIs are given since there are 4 floors in the building.



M2.2: Verifier

The following properties are checked.

M2.2.1

Property	E<> floorUI.Arrived
Description	The elevator will arrive at this floor at some moment.
Result	Passed

M2.2.2

Property	E<> floorUI.DoorOpen
Description	The door will open at some moment.
Result	Passed

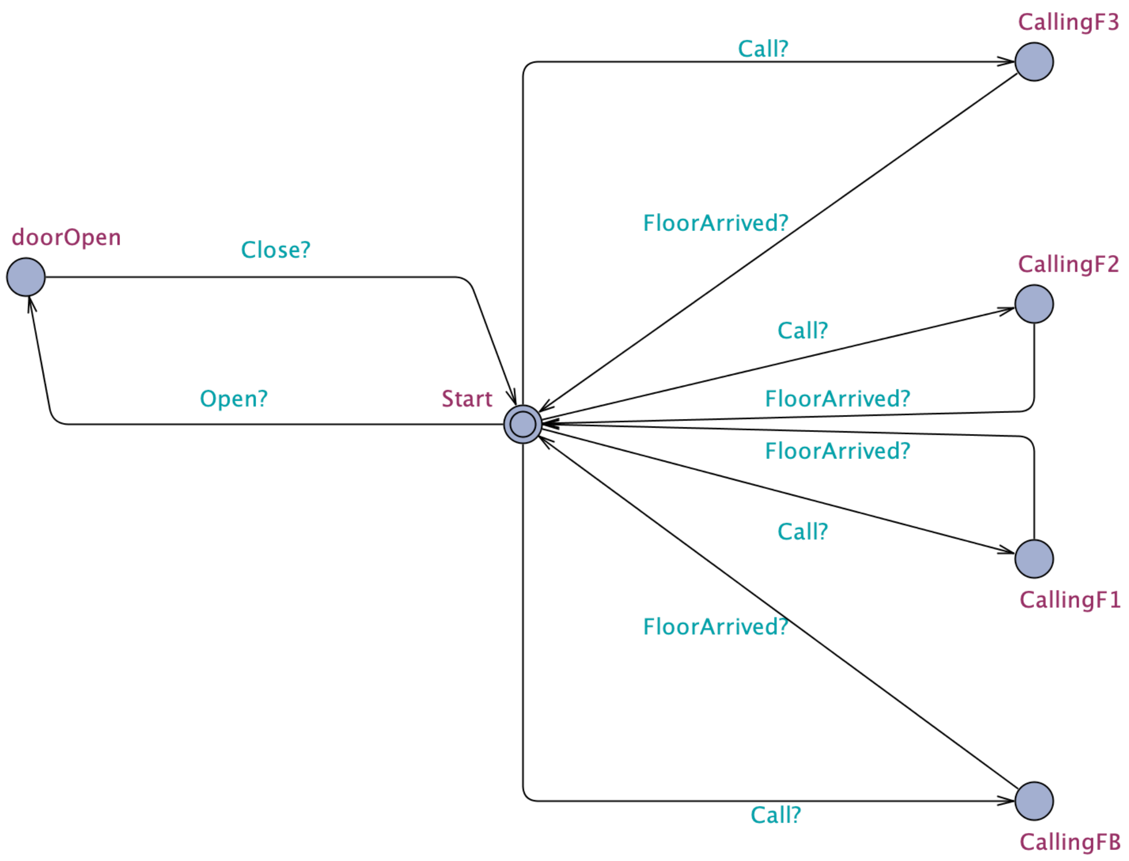
M2.2.3

Property	E<> floorUI.Called
Description	Calls can be made for some time needed.
Result	Passed

M3: CarUI

M3.1: Simulator

Users can assign target floor to go through the CarUI. The door status can also be seen in this simulator.



M3.2: Verifier

The following properties are checked.

M3.2.1

Property	E<> carUI.CallingF2
Description	F2 can be assigned as the target floor.
Result	Passed

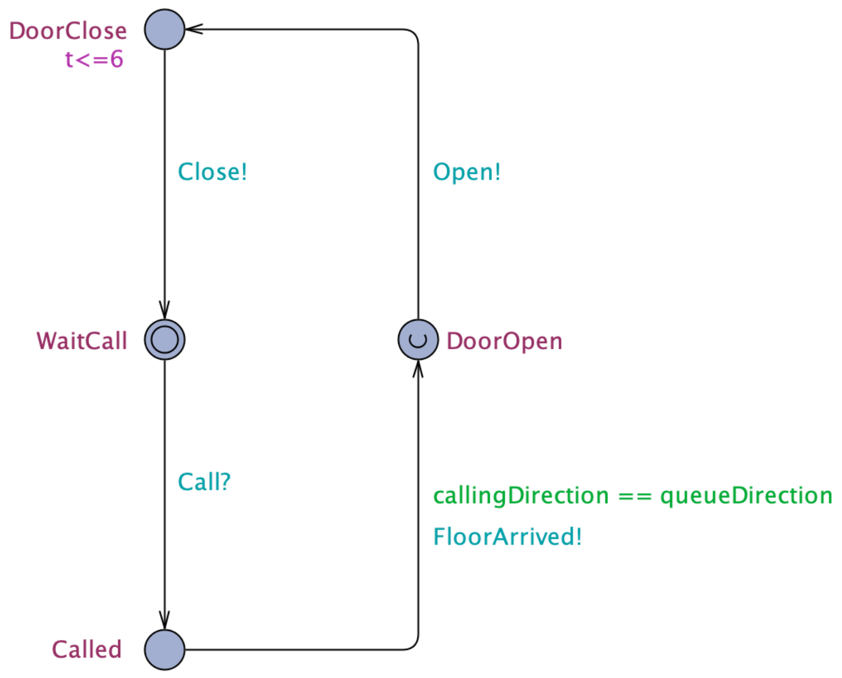
M3.2.2

Property	E<> carUI.doorOpen
Description	The door will open at the target floor.
Result	Passed

M4: Controller

M4.1: Simulator

The controller receives message from user calls and sends message to the door.



M4.2: Verifier

The following properties are checked.

M4.2.1

Property	E<> controller.Called
Description	Calls made by users can be received.
Result	Passed

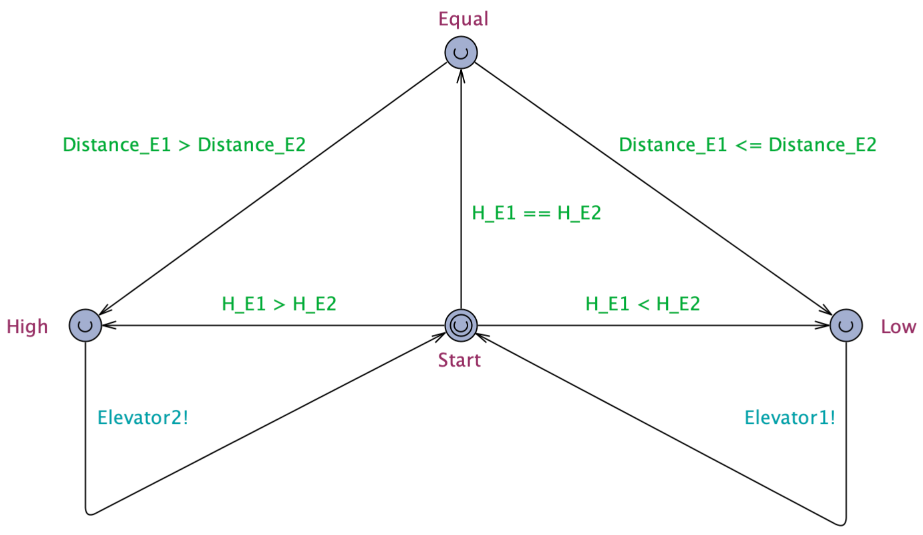
M4.2.2

Property	E<> controller.DoorClose
Description	The door will close after at most 6 seconds.
Result	Passed

M5: Scheduler

M5.1: Simulator

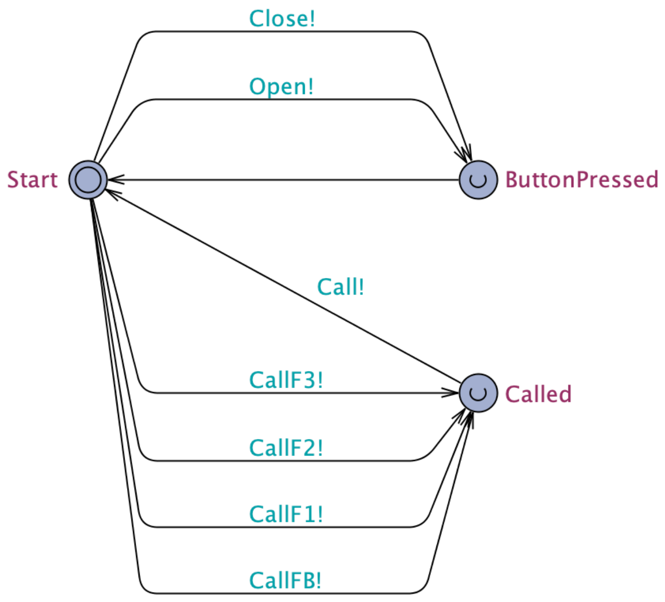
The scheduler compares the dispatch hierarchy of two elevators and sending message to elevators.



M6: User

M6.1: Simulator

Users can control the door status (Close or Open) and assign floors to go.



M6.2: Verifier

The following properties are checked.

M6.2.1

Property	E<> user.Called
Description	Calls can be made by users.

Result	Passed

M6.2.2

Property	E<> user.ButtonPressed
Description	Users can select floor to go and make calls to the system.
Result	Passed