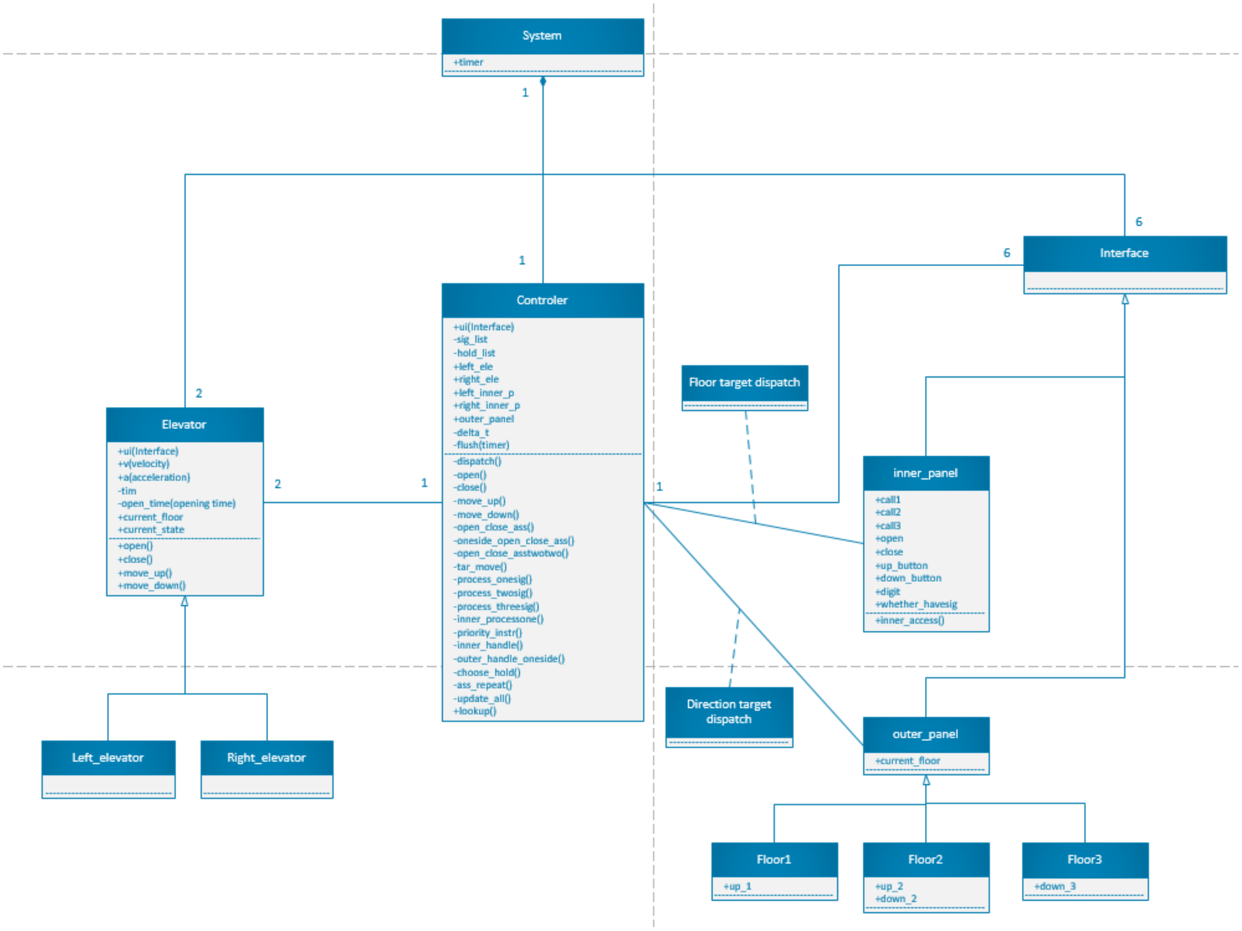
Validation

System Architecture



T1 Unit Test

T1.1 Controller Test

T1.1.1 Test scheduler()

```
function scheduler(obj,signal)
    obj.sig_list(signal) = 1;
end
```

- Coverage Criteria: Statement Coverage
- Test case

	Test Case T1.1.1
Coverage Item	Tcover 1.1.1
Input	signal = 1
State	sig_list(1) == -1
Expected Output	sig_list(1) == 1

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

T1.1.2 Test open_close_ass()

```
function a = open_close_ass(obj,fir,c_position,c_state)
    a = 0;
    if fir == 1 % Tcover 1.1.2.1
        fl = 3;
        po = 467;
    elseif fir == 2 || fir == 3 % Tcover 1.1.2.2
        fl = 2;
        po = 352;
    elseif fir == 4 % Tcover 1.1.2.3
        fl = 1;
        po = 236;
    end

    % Left is in there -open
    if c_position(1) == po && (c_state(1) == 3) && obj.hold_list(fir) == 0
        obj.open(0,fl,obj.delta_t,fir)
        a = 1;
        obj.hold_list(fir) = 0;
        return; % Tcover 1.1.2.4
    % Right is in there -open
    elseif c_position(2) == po && (c_state(2) == 3) && obj.hold_list(fir) == 1
        obj.open(1,fl,obj.delta_t,fir)
        a = 1;
        obj.hold_list(fir) = 1;
        return; % Tcover 1.1.2.5
    % Left -Max-open
    elseif c_position(1) == po && (c_state(1) == 5) && obj.hold_list(fir) == 0
        if obj.sig_list(6) == 1 % Tcover 1.1.2.6
            a = 1;
            obj.left_ele.tim = 0;
            obj.left_ele.current_state = 4;
            return;
        end
        a = 1;
        obj.left_ele.tim = obj.left_ele.tim + 0.2;
        obj.hold_list(fir) = 0;
        if obj.left_ele.tim >= obj.left_ele.open_tim % Tcover 1.1.2.7
            obj.left_ele.current_state = 4;
            obj.left_ele.tim = 0;
        end
        return;
    % Left -close
    elseif c_position(1) == po && (c_state(1) == 4) && obj.hold_list(fir) == 0
        a = 1;
        obj.hold_list(fir) = 0;
        obj.close(0,fl,obj.delta_t,fir);
        return; % Tcover 1.1.2.8
    % Right -Max-open
    elseif c_position(2) == po && (c_state(2) == 5) && obj.hold_list(fir) == 1
        if obj.sig_list(14) == 1 % Tcover 1.1.2.9
            a = 1;
            obj.right_ele.tim = 0;
            obj.right_ele.current_state = 4;
            return;
        end
        a = 1;
        obj.right_ele.tim = obj.right_ele.tim + 0.2;
        obj.hold_list(fir) = 1;
        if obj.right_ele.tim >= obj.right_ele.open_tim % Tcover 1.1.2.10
            obj.right_ele.current_state = 4;
            obj.right_ele.tim = 0;
        end
        return;
    % Right -close
    elseif c_position(2) == po && (c_state(2) == 4) && obj.hold_list(fir) == 1
        a = 1;
        obj.hold_list(fir) = 1;
        obj.close(1,fl,obj.delta_t,fir)
        return; % Tcover 1.1.2.11
    % Left is in there -open
    elseif c_position(1) == po && (c_state(1) == 0) && obj.hold_list(fir) == 0
        obj.left_ele.current_state = 3;
        obj.open(0,fl,obj.delta_t,fir)
        a = 1;
        obj.hold_list(fir) = 0;
        return; % Tcover 1.1.2.12
    % Right is in there -open
    elseif c_position(2) == po && (c_state(2) == 0) && obj.hold_list(fir) == 1
        obj.right_ele.current_state = 3;
        obj.open(1,fl,obj.delta_t,fir)
        a = 1;
        obj.hold_list(fir) = 1;
        return; % Tcover 1.1.2.13
    end
end
```

- Coverage Criteria: Branch Coverage
- Test case

	Test Case T1.1.2.1
Coverage Item	Tcover 1.1.2.1, Tcover 1.1.2.4

Input	fir = 1
	c_position = [467,250]
	c_state = [3,1]

	Test Case T1.1.2.1
State	hold_list(fir) == 0
Expected Output	a == 1

	Test Case T1.1.2.2
Coverage Item	Tcover 1.1.2.2, Tcover 1.1.2.5

Input	fir = 2
	c_position = [250,352]
	c_state = [1,3]

State	hold_list(fir) == 1
Expected Output	a == 1

	Test Case T1.1.2.3
Coverage Item	Tcover 1.1.2.3, Tcover 1.1.2.6

Input	fir = 4
	c_position = [236,352]
	c_state = [5,3]

State	hold_list(fir) == 0 sig_list(6) == 1
Expected Output	a == 1

	Test Case T1.1.2.4
Coverage Item	Tcover 1.1.2.1, Tcover 1.1.2.7

Input	fir = 1
	c_position = [467,352]
	c_state = [5,3]

State	hold_list(fir) == 0 left_ele.tim == left_ele.open_tim
Expected Output	a == 1

	Test Case T1.1.2.5
Coverage Item	Tcover 1.1.2.1, Tcover 1.1.2.8

	Test Case T1.1.2.5

Input	fir = 1
	c_position = [467,352]
	c_state = [4,3]

State	hold_list(fir) == 0
Expected Output	a == 1

	Test Case T1.1.2.6
Coverage Item	Tcover 1.1.2.1, Tcover 1.1.2.9

Input	fir = 1
	c_position = [353,467]
	c_state = [4,5]

State	hold_list(fir) == 1 sig_list(14) == 1
Expected Output	a == 1

	Test Case T1.1.2.7
Coverage Item	Tcover 1.1.2.1, Tcover 1.1.2.10

Input	fir = 1
	c_position = [353,467]
	c_state = [4,5]

State	hold_list(fir) == 1 right_ele.tim == right_ele.open_tim
Expected Output	a == 1

	Test Case T1.1.2.8
Coverage Item	Tcover 1.1.2.1, Tcover 1.1.2.11

Input	fir = 1
	c_position = [353,467]
	c_state = [3,4]

State	hold_list(fir) == 1
Expected Output	a == 1

	Test Case T1.1.2.9

	Test Case T1.1.2.9
Coverage Item	Tcover 1.1.2.1, Tcover 1.1.2.12

Input	fir = 1
	c_position = [467,333]
	c_state = [0,4]

State	hold_list(fir) == 0
Expected Output	a == 1

	Test Case T1.1.2.10
Coverage Item	Tcover 1.1.2.1, Tcover 1.1.2.13

Input	fir = 1
	c_position = [333,467]
	c_state = [1,0]

State	hold_list(fir) == 1
Expected Output	a == 1

- Test coverage: 13/13=100%
- Test Result: 10 passed

T1.1.3 Test oneside_open_close_ass()

```
function a = oneside_open_close_ass(obj,fir,c_position,c_state,side)
    % Side == 1 -> left, == 2 -> right
    a = 0;
    if fir == -1 % Tcover 1.1.3.1
        return;
    end
    if fir == 7 || fir == 10 || fir == 1 % Tcover 1.1.3.2
        fl = 3;
        po = 467;
    elseif fir == 8 || fir == 11 || fir == 2 || fir == 3 % Tcover 1.1.3.3
        fl = 2;
        po = 352;
    elseif fir == 9 || fir == 12 || fir == 4 % Tcover 1.1.3.4
        fl = 1;
        po = 236;
    end

    % -open
    if c_position(side) == po && (c_state(side) == 3) && obj.hold_list(fir) == side - 1
        a = 1;
        obj.hold_list(fir) = side - 1;
        obj.open(side - 1,fl,obj.delta_t,fir);
        return; % Tcover 1.1.3.5
    % -open-Max
    elseif c_position(side) == po && c_state(side) == 5 && obj.hold_list(fir) ==
side - 1
        a = 1;
        obj.hold_list(fir) = side - 1;
        if side == 1
            if obj.sig_list(6) == 1 % Tcover 1.1.3.6
                a = 1;
                obj.left_ele.tim = 0;
                obj.left_ele.current_state = 4;
                return;
            end
            obj.left_ele.tim = obj.left_ele.tim + 0.2;
            if obj.left_ele.tim >= obj.left_ele.open_tim % Tcover 1.1.3.7
                obj.left_ele.current_state = 4;
                obj.left_ele.tim = 0;
            end
        elseif side == 2
            if obj.sig_list(14) == 1 % Tcover 1.1.3.8
                a = 1;
                obj.right_ele.tim = 0;
                obj.right_ele.current_state = 4;
                return;
            end
            obj.right_ele.tim = obj.right_ele.tim + 0.2;
            if obj.right_ele.tim >= obj.right_ele.open_tim % Tcover 1.1.3.9
                obj.right_ele.current_state = 4;
                obj.right_ele.tim = 0;
            end
        end
        return;
    % -close
    elseif c_position(side) == po && (c_state(side) == 4 || c_state(side) == 5) &&
obj.hold_list(fir) == side - 1 % Tcover 1.1.3.10
        a = 1;
        obj.hold_list(fir) = side - 1;
        obj.close(side - 1,fl,obj.delta_t,fir);
        return;
    % In there -open
    elseif c_position(side) == po && (c_state(side) == 0) && obj.hold_list(fir) ==
side - 1
        a = 1;
        obj.hold_list(fir) = side - 1;
        if side == 1 % Tcover 1.1.3.11
            obj.left_ele.current_state = 3;
        elseif side == 2 % Tcover 1.1.3.12
            obj.right_ele.current_state = 3;
        end
        obj.open(side - 1,fl,obj.delta_t,fir);
        return;
        end
    end
```

- Coverage Criteria: Branch Coverage
- Test case

	Test Case T1.1.3.1
Coverage Item	Tcover 1.1.3.1
Input	fir = -1
State	----------------
Expected Output	a == 0

	Test Case T1.1.3.2
Coverage Item	Tcover 1.1.3.2, Tcover 1.1.3.5

Input	fir = 7
	side = 1
	c_position = [467,352]
	c_state = [3,2]

State	hold_list(fir) == side - 1
Expected Output	a == 1

	Test Case T1.1.3.3
Coverage Item	Tcover 1.1.3.3, Tcover 1.1.3.6

Input	fir = 8
	side = 1
	c_position = [352,236]
	c_state = [5,3]

State	hold_list(fir) == side - 1 sig_list(6) == 1
Expected Output	a == 1

	Test Case T1.1.3.4
Coverage Item	Tcover 1.1.3.4, Tcover 1.1.3.7

	Test Case T1.1.3.4

Input	fir = 9
	side = 1
	c_position = [236,352]
	c_state = [5,3]

State	hold_list(fir) == side - 1 left_ele.tim == left_ele.open_tim
Expected Output	a == 1

	Test Case T1.1.3.5
Coverage Item	Tcover 1.1.3.2, Tcover 1.1.3.8

Input	fir = 7
	side = 2
	c_position = [333,467]
	c_state = [3,5]

State	hold_list(fir) == side - 1 sig_list(14) == 1
Expected Output	a == 1

	Test Case T1.1.3.6
Coverage Item	Tcover 1.1.3.2, Tcover 1.1.3.9

Input	fir = 7
	side = 2
	c_position = [353,467]
	c_state = [4,5]

State	hold_list(fir) == side - 1 right_ele.tim == right_ele.open_tim
Expected Output	a == 1

	Test Case T1.1.3.7
Coverage Item	Tcover 1.1.3.2, Tcover 1.1.3.10

Input	fir = 7
	side = 1
	c_position = [467,333]
	c_state = [4,5]

State	hold_list(fir) == side - 1

	Test Case T1.1.3.7
Expected Output	a == 1

	Test Case T1.1.3.8
Coverage Item	Tcover 1.1.3.2, Tcover 1.1.3.11

Input	fir = 7
	side = 1
	c_position = [467,333]
	c_state = [0,4]

State	hold_list(fir) == side - 1
Expected Output	a == 1

	Test Case T1.1.3.9
Coverage Item	Tcover 1.1.3.2, Tcover 1.1.3.12

Input	fir = 7
	side = 2
	c_position = [333,467]
	c_state = [1,0]

State	hold_list(fir) == side - 1
Expected Output	a == 1

- Test coverage: 12/12=100%
- Test Result: 9 passed

T1.1.4 Test open_close_asstwotwo()

```
function [a,b] = open_close_asstwotwo(obj,c_position,c_state)
    a = 0;
    b = 0;
    fl = 2;
    po = 352;

    % Set corresponding elevator -> command 2 or 3
    if obj.hold_list(2) == 0 % Tcover 1.1.4.1
        left_receive = 2;
        right_receive = 3;
    elseif obj.hold_list(2) == 1 % Tcover 1.1.4.2
        right_receive = 2;
        left_receive = 3;
    end
    if obj.hold_list(3) == 0 % Tcover 1.1.4.3
```

```
        left_receive = 3;
        right_receive = 2;
    elseif obj.hold_list(3) == 1 % Tcover 1.1.4.4
        right_receive = 3;
        left_receive = 2;
    end
    if left_receive == 2 % Tcover 1.1.4.5
        obj.hold_list(2) = 0;
    elseif left_receive == 3 % Tcover 1.1.4.6
        obj.hold_list(3) = 0;
    end
    if right_receive == 2 % Tcover 1.1.4.7
        obj.hold_list(2) = 1;
    elseif right_receive == 3 % Tcover 1.1.4.8
        obj.hold_list(3) = 1;
    end

    % Left is in there -open
    if c_position(1) == po && (c_state(1) == 3) % Tcover 1.1.4.9
        obj.open(0,fl,obj.delta_t,2)
        a = 1;
    end
    % Right is in there -open
    if c_position(2) == po && (c_state(2) == 3) % Tcover 1.1.4.10
        obj.open(1,fl,obj.delta_t,2)
        b = 1;
    end
    % Left -close
    if c_position(1) == po && c_state(1) == 5
        a = 1;
        obj.left_ele.tim = obj.left_ele.tim + 0.2;
        if obj.left_ele.tim >= obj.left_ele.open_tim % Tcover 1.1.4.11
            obj.left_ele.current_state = 4;
            obj.left_ele.tim = 0;
        end
        if obj.sig_list(6) == 1 % Tcover 1.1.4.12
            obj.left_ele.tim = 0;
            obj.left_ele.current_state = 4;
        end
    elseif c_position(1) == po && c_state(1) == 4 % Tcover 1.1.4.13
        obj.close(0,fl,obj.delta_t,left_receive)
        a = 1;
    end
    % Right -close
    if c_position(2) == po && c_state(2) == 5
        b = 1;
        obj.right_ele.tim = obj.right_ele.tim + 0.2;
        if obj.right_ele.tim >= obj.right_ele.open_tim % Tcover 1.1.4.14
            obj.right_ele.current_state = 4;
            obj.right_ele.tim = 0;
        end
        if obj.sig_list(14) == 1 % Tcover 1.1.4.15
```

```
            obj.right_ele.tim = 0;
            obj.right_ele.current_state = 4;
        end
    elseif c_position(2) == po && c_state(2) == 4 % Tcover 1.1.4.16
        obj.close(1,fl,obj.delta_t,right_receive)
        b = 1;
    end
    % Left is in there -open
    if c_position(1) == po && (c_state(1) == 0) % Tcover 1.1.4.17
        obj.left_ele.current_state = 3;
        obj.open(0,fl,obj.delta_t,2)
        a = 1;
    end
    % Right is in there -open
    if c_position(2) == po && (c_state(2) == 0) % Tcover 1.1.4.18
        obj.right_ele.current_state = 3;
        obj.open(1,fl,obj.delta_t,2)
        b = 1;
    end

    if [a,b] == [0,0] % Tcover 1.1.4.19
        tar1 = obj.dispatch_oneside(left_receive,1);
        tar2 = obj.dispatch_oneside(right_receive,2);
        obj.tar_move(tar1);
        obj.tar_move(tar2);
    elseif [a,b] == [0,1] % Tcover 1.1.4.20
        tar1 = obj.dispatch_oneside(left_receive,1);
        obj.tar_move(tar1);
    elseif [a,b] == [1,0] % Tcover 1.1.4.21
        tar2 = obj.dispatch_oneside(right_receive,2);
        obj.tar_move(tar2);
    end
end
```

- Coverage Criteria: Branch Coverage
- Test case

	Test Case T1.1.4.1
Coverage Item	Tcover 1.1.4.1, Tcover 1.1.4.3, Tcover 1.1.4.5, Tcover 1.1.4.7, Tcover 1.1.4.9, Tcover 1.1.4.10

Input	c_position = [352,352]
	c_state = [3,3]

State	hold_list(2) == 0 hold_list(3) == 1
Expected Output	a == 1

	Test Case T1.1.4.2
Coverage Item	Tcover 1.1.4.2, Tcover 1.1.4.4, Tcover 1.1.4.6, Tcover 1.1.4.8, Tcover 1.1.4.11, Tcover 1.1.4.14

Input	c_position = [352,352]
	c_state = [5,5]

State	hold_list(2) == 1 hold_list(3) == 0
Expected Output	a == 1

	Test Case T1.1.4.3
Coverage Item	Tcover 1.1.4.1, Tcover 1.1.4.3, Tcover 1.1.4.5, Tcover 1.1.4.7, Tcover 1.1.4.12, Tcover 1.1.4.15

Input	c_position = [352,352]
	c_state = [5,5]

State	hold_list(2) == 1 hold_list(3) == 0 sig_list(6) == 1 sig_list(14) == 1
Expected Output	a == 1

	Test Case T1.1.4.4
Coverage Item	Tcover 1.1.4.1, Tcover 1.1.4.3, Tcover 1.1.4.5, Tcover 1.1.4.7, Tcover 1.1.4.13, Tcover 1.1.4.16

Input	c_position = [352,352]
	c_state = [5,5]

State	hold_list(2) == 0 hold_list(3) == 1 sig_list(6) == 1 sig_list(14) == 1
Expected Output	a == 1

	Test Case T1.1.4.5

	Test Case T1.1.4.5
Coverage Item	Tcover 1.1.4.1, Tcover 1.1.4.3, Tcover 1.1.4.5, Tcover 1.1.4.7, Tcover 1.1.4.17, Tcover 1.1.4.18

Input	c_position = [352,352]
	c_state = [5,5]

State	hold_list(2) == 0 hold_list(3) == 1 left_ele.tim = left_ele.open_tim right_ele.tim == right_ele.open_tim
Expected Output	a == 1

	Test Case T1.1.4.6
Coverage Item	Tcover 1.1.4.19

Input	c_position = [333,333]
	c_state = [1,1]

State	hold_list(2) == 0 hold_list(3) == 1
Expected Output	a == 0

	Test Case T1.1.4.7
Coverage Item	Tcover 1.1.4.20

Input	c_position = [333,352]
	c_state = [1,3]

State	hold_list(2) == 0 hold_list(3) == 1
Expected Output	a == 0

	Test Case T1.1.4.8
Coverage Item	Tcover 1.1.4.21

Input	c_position = [352,333]
	c_state = [3,1]

State	hold_list(2) == 0 hold_list(3) == 1

	Test Case T1.1.4.8
Expected Output	a == 1

- Test coverage: 21/21=100%
- Test Result: 8 passed

T1.1.5 Test tar_move()

```
function tar_move(obj,tar)
    if tar == [0,1] % Tcover 1.1.5.1
        obj.left_ele.current_state = 1;
        obj.move_up(0,obj.delta_t);
    elseif tar == [0,2] % Tcover 1.1.5.2
        obj.left_ele.current_state = 2;
        obj.move_down(0,obj.delta_t);
    elseif tar == [1,1] % Tcover 1.1.5.3
        obj.right_ele.current_state = 1;
        obj.move_up(1,obj.delta_t);
    elseif tar == [1,2] % Tcover 1.1.5.4
        obj.right_ele.current_state = 2;
        obj.move_down(1,obj.delta_t);
    end
end
```

- Coverage Criteria: Branch Coverage
- Test case

	Test Case T1.1.5.1
Coverage Item	Tcover 1.1.5.1
Input	tar = [0,1]
State	---------
Expected Output	left_ele.current_state == 1

	Test Case T1.1.5.2
Coverage Item	Tcover 1.1.5.2
Input	tar = [0,2]
State	---------
Expected Output	left_ele.current_state == 2

	Test Case T1.1.5.3

	Test Case T1.1.5.3
Coverage Item	Tcover 1.1.5.3
Input	tar = [1,1]
State	---------
Expected Output	right_ele.current_state == 1

	Test Case T1.1.5.4
Coverage Item	Tcover 1.1.5.4
Input	tar = [1,2]
State	---------
Expected Output	right_ele.current_state == 2

- Test coverage: 4/4=100%
- Test Result: 4 passed

T1.1.6 Test process_onesig()

```
function process_onesig(obj,fir,c_position,c_state)
    see = obj.open_close_ass(fir,c_position,c_state);

    % Schedule one elevator to react
    if see == 0 % Tcover 1.1.6.1
        tar = obj.dispatch(fir);
        obj.tar_move(tar);
    end
end
```

- Coverage Criteria: Branch Coverage
- Test case

	Test Case T1.1.6.1
Coverage Item	Tcover 1.1.6.1

Input	fir = 1
	c_position = [277,277]
	c_state = [2,2]

State	---------
Expected Output	right_ele.current_state == 0

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

T1.1.7 Test process_twosig()

```
function process_twosig(obj,fir,sec,c_position,c_state)
    tar1 = [-1,-1];
    tar2 = [-1,-1];

    see1 = obj.open_close_ass(fir,c_position,c_state);
    see2 = obj.open_close_ass(sec,c_position,c_state);

    % The condition that one elevator hold two instructions
    if obj.hold_list(fir) == obj.hold_list(sec)
        working = obj.hold_list(fir);
        if working == 0 && (obj.left_ele.current_state == 3 ||
obj.left_ele.current_state == 4 || obj.left_ele.current_state == 5) % Tcover 1.1.7.1
            return;
        elseif working == 1 && (obj.right_ele.current_state == 3 ||
obj.right_ele.current_state == 4 || obj.right_ele.current_state == 5)
            return; % Tcover 1.1.7.2
        end
    end

    % Schedule one elevator to react
    if see1 == 0 % Tcover 1.1.7.3
        tar1 = obj.dispatch(fir);
    end
    if see2 == 0 % Tcover 1.1.7.4
        tar2 = obj.dispatch(sec);
    end

    if tar1 == tar2 % Tcover 1.1.7.5
        obj.tar_move(tar1);
    else % Tcover 1.1.7.6
        obj.tar_move(tar1);
        obj.tar_move(tar2);
    end
end
```

- Coverage Criteria: Branch Coverage
- Test case

	Test Case T1.1.7.1
Coverage Item	Tcover 1.1.7.1

	Test Case T1.1.7.1

Input	fir = 1
	sec = 2
	c_position = [277,277]
	c_state = [2,2]

State	hold_list(fir) == hold_list(sec) == 0
Expected Output	left_ele.current_state == 1

	Test Case T1.1.7.2
Coverage Item	Tcover 1.1.7.2

Input	fir = 1
	sec = 2
	c_position = [277,277]
	c_state = [2,2]

State	hold_list(fir) == hold_list(sec) == 1
Expected Output	left_ele.current_state == 0

	Test Case T1.1.7.3
Coverage Item	Tcover 1.1.7.3, Tcover 1.1.7.4, Tcover 1.1.7.5

Input	fir = 1
	sec = 2
	c_position = [467,352]
	c_state = [0,0]

State	---------
Expected Output	left_ele.current_state == 1

	Test Case T1.1.7.4
Coverage Item	Tcover 1.1.7.3, Tcover 1.1.7.4, Tcover 1.1.7.6

Input	fir = 1
	sec = 2
	c_position = [466,353]
	c_state = [0,0]

State	---------
Expected Output	left_ele.current_state == 1

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

T1.1.8 Test process_threesig()

```
function process_threesig(obj,fir,sec,thi,c_position,c_state)
    tar1 = [-1,-1];
    tar2 = [-1,-1];
    tar3 = [-1,-1];

    see1 = obj.open_close_ass(fir,c_position,c_state);
    see2 = obj.open_close_ass(sec,c_position,c_state);
    see3 = obj.open_close_ass(thi,c_position,c_state);

    % The condition that one elevator hold two instructions
    if obj.hold_list(fir) == obj.hold_list(sec)
        working = obj.hold_list(fir);
        if working == 0 && (obj.left_ele.current_state == 3 || obj.left_ele.current_state == 4 || obj.left_ele.current_state == 5) % Tcover 1.1.8.1
            if see3 == 0 % Tcover 1.1.8.2
                tar3 = obj.dispatch_oneside(thi,2);
                obj.tar_move(tar3);
            end
            return;
        elseif working == 1 && (obj.right_ele.current_state == 3 || obj.right_ele.current_state == 4 || obj.right_ele.current_state == 5) % Tcover 1.1.8.3
            if see3 == 0 % Tcover 1.1.8.4
                tar3 = obj.dispatch_oneside(thi,1);
                obj.tar_move(tar3);
            end
            return;
        end
    elseif obj.hold_list(sec) == obj.hold_list(thi) % Tcover 1.1.8.5
        working = obj.hold_list(sec);
        if working == 0 && (obj.left_ele.current_state == 3 || obj.left_ele.current_state == 4 || obj.left_ele.current_state == 5) % Tcover 1.1.8.6
            if see3 == 0 % Tcover 1.1.8.7
                tar3 = obj.dispatch_oneside(fir,2);
                obj.tar_move(tar3);
            end
            return;
        elseif working == 1 && (obj.right_ele.current_state == 3 || obj.right_ele.current_state == 4 || obj.right_ele.current_state == 5) % Tcover 1.1.8.8
            if see3 == 0 % Tcover 1.1.8.9
                tar3 = obj.dispatch_oneside(fir,1);
                obj.tar_move(tar3);
            end
            return;
        end
    elseif obj.hold_list(fir) == obj.hold_list(thi) % Tcover 1.1.8.10
        working = obj.hold_list(thi);
        if working == 0 && (obj.left_ele.current_state == 3 || obj.left_ele.current_state == 4 || obj.left_ele.current_state == 5) % Tcover 1.1.8.11
            if see3 == 0 % Tcover 1.1.8.12
                tar3 = obj.dispatch_oneside(sec,2);
                obj.tar_move(tar3);
            end
            return;
        elseif working == 1 && (obj.right_ele.current_state == 3 || obj.right_ele.current_state == 4 || obj.right_ele.current_state == 5) % Tcover 1.1.8.13
            if see3 == 0 % Tcover 1.1.8.14
                tar3 = obj.dispatch_oneside(sec,1);
                obj.tar_move(tar3);
            end
            return;
        end
    end

    % Schedule one elevator to react
    if see1 == 0 % Tcover 1.1.8.15
        tar1 = obj.dispatch(fir);
    end
    if see2 == 0 % Tcover 1.1.8.16
        tar2 = obj.dispatch(sec);
    end
    if see3 == 0 % Tcover 1.1.8.17
        tar3 = obj.dispatch(thi);
    end

    if (tar1 == tar2) & (tar2 == tar3) % Tcover 1.1.8.18
        obj.tar_move(tar1);
    elseif (tar1 == tar2) % Tcover 1.1.8.19
        obj.tar_move(tar2);
        obj.tar_move(tar3);
    elseif (tar2 == tar3) % Tcover 1.1.8.20
        obj.tar_move(tar1);
        obj.tar_move(tar3);
    elseif (tar1 == tar3) % Tcover 1.1.8.21
        obj.tar_move(tar1);
        obj.tar_move(tar2);
    else % Tcover 1.1.8.22
        obj.tar_move(tar1);
        obj.tar_move(tar2);
        obj.tar_move(tar3);
    end
end
```

- Coverage Criteria: Branch Coverage

- Test case

	Test Case T1.1.8.1
Coverage Item	Tcover 1.1.8.1, Tcover 1.1.8.2, Tcover 1.1.8.15, Tcover 1.1.8.16, Tcover 1.1.8.17, Tcover 1.1.8.22

Input	fir = 1
	sec = 2
	thi = 3
	c_position = [277,277]
	c_state = [2,2]

State	hold_list(fir) == 0 hold_list(sec) == 0 hold_list(thi) == 1
Expected Output	left_ele.current_state == 1

	Test Case T1.1.8.2
Coverage Item	Tcover 1.1.8.3, Tcover 1.1.8.4, Tcover 1.1.8.18

Input	fir = 4
	sec = 2
	thi = 3
	c_position = [352,277]
	c_state = [3,2]

State	hold_list(fir) == 0 hold_list(sec) == 1 hold_list(thi) == 1
Expected Output	left_ele.current_state == 0

	Test Case T1.1.8.3
Coverage Item	Tcover 1.1.8.5, Tcover 1.1.8.6, Tcover 1.1.8.7, Tcover 1.1.8.19

Input	fir = 1
	sec = 2
	thi = 3
	c_position = [450,352]
	c_state = [1,0]

State	hold_list(fir) == 0 hold_list(sec) == 0 hold_list(thi) == 1

	Test Case T1.1.8.3
Expected Output	left_ele.current_state == 1

	Test Case T1.1.8.4
Coverage Item	Tcover 1.1.8.8, Tcover 1.1.8.9, Tcover 1.1.8.20

Input	fir = 1
	sec = 3
	thi = 4
	c_position = [236,450]
	c_state = [2,2]

State	hold_list(fir) == 0 hold_list(sec) == 0 hold_list(thi) == 1
Expected Output	left_ele.current_state == 1

	Test Case T1.1.8.5
Coverage Item	Tcover 1.1.8.10, Tcover 1.1.8.11, Tcover 1.1.8.12, Tcover 1.1.8.21

Input	fir = 2
	sec = 3
	thi = 4
	c_position = [236,236]
	c_state = [1,1]

State	hold_list(fir) == 0 hold_list(sec) == 0 hold_list(thi) == 1
Expected Output	left_ele.current_state == 1

	Test Case T1.1.8.6
Coverage Item	Tcover 1.1.8.13, Tcover 1.1.8.14

Input	fir = 2
	sec = 3
	thi = 4
	c_position = [467,467]
	c_state = [2,2]

State	hold_list(fir) == 0 hold_list(sec) == 1 hold_list(thi) == 1

	Test Case T1.1.8.6
Expected Output	left_ele.current_state == 1

- Test coverage: 22/22=100%
- Test Result: 6 passed

T1.1.9 Test outer_handle_oneside()

```
% Let side(1->left,2->right) elevator to handle some outer signal
function outer_handle_oneside(obj,side,out_sig,c_position,c_state)
    if side == 1 % Tcover 1.1.9.1
        if (obj.sig_list(10) == 1 && obj.hold_list(1) ~= 0) || (c_position(2) == 467
&& c_state(2) ~= 2) % Tcover 1.1.9.2
            out_sig(1) = 0;
        end
        if (obj.sig_list(12) == 1 && obj.hold_list(4) ~= 0) || (c_position(2) == 236
&& c_state(2) ~= 1) % Tcover 1.1.9.3
            out_sig(4) = 0;
        end
        if c_position(2) == 352 && c_state(2) ~= 2 && obj.hold_list(3) ~= 1 &&
obj.hold_list(4) ~= 1 % Tcover 1.1.9.4
            out_sig(2) = 0;
        end
        if c_position(2) == 352 && c_state(2) ~= 1 && obj.hold_list(1) ~= 1 &&
obj.hold_list(2) ~= 1 % Tcover 1.1.9.5
            out_sig(3) = 0;
        end
    elseif side == 2 % Tcover 1.1.9.6
        if (obj.sig_list(7) == 1 && obj.hold_list(1) ~= 1) || (c_position(1) == 467
&& c_state(1) ~= 2) % Tcover 1.1.9.7
            out_sig(1) = 0;
        end
        if (obj.sig_list(9) == 1 && obj.hold_list(4) ~= 1) || (c_position(1) == 236
&& c_state(1) ~= 1) % Tcover 1.1.9.8
            out_sig(4) = 0;
        end
        if c_position(1) == 352 && c_state(1) ~= 2 && obj.hold_list(3) ~= 0 &&
obj.hold_list(4) ~= 0 % Tcover 1.1.9.9
            out_sig(2) = 0;
        end
        if c_position(1) == 352 && c_state(1) ~= 1 && obj.hold_list(1) ~= 0 &&
obj.hold_list(2) ~= 0 % Tcover 1.1.9.10
            out_sig(3) = 0;
        end
    end
    com = obj.priority_instr(out_sig,2,c_position,c_state,side);
    see = oneside_open_close_ass(obj,com,c_position,c_state,side);
    if com ~= -1 && see == 0 % Tcover 1.1.9.11
        tar = obj.dispatch_oneside(com,side);
        obj.tar_move(tar);
    end
end
```

- Coverage Criteria: Branch Coverage
- Test case

	Test Case T1.1.9.1
Coverage Item	Tcover 1.1.9.1, Tcover 1.1.9.2, Tcover 1.1.9.11

Input	side = 1
	out_sig = [0,0,0,0,0]
	c_position = [236,467]
	c_state = [1,2]

State	hold_list(1) == -1 hold_list(2) == -1 hold_list(3) == -1 hold_list(4) == -1
Expected Output	left_ele.current_state == 0

	Test Case T1.1.9.2
Coverage Item	Tcover 1.1.9.1, Tcover 1.1.9.3

Input	side = 1
	out_sig = [1,0,1,0,0]
	c_position = [352,467]
	c_state = [2,2]

State	hold_list(1) == 0 hold_list(2) == -1 hold_list(3) == -1 hold_list(4) == -1
Expected Output	left_ele.current_state == 1

	Test Case T1.1.9.3
Coverage Item	Tcover 1.1.9.1, Tcover 1.1.9.4

Input	side = 2
	out_sig = [0,0,0,1,0]
	c_position = [236,467]
	c_state = [1,2]

	Test Case T1.1.9.3
State	hold_list(1) == -1 hold_list(2) == 1 hold_list(3) == -1 hold_list(4) == -1
Expected Output	left_ele.current_state == 0

	Test Case T1.1.9.4
Coverage Item	Tcover 1.1.9.1, Tcover 1.1.9.5

Input	side = 2
	out_sig = [0,1,0,0,1]
	c_position = [236,467]
	c_state = [1,2]

State	hold_list(1) == -1 hold_list(2) == 0 hold_list(3) == -1 hold_list(4) == 1
Expected Output	left_ele.current_state == 0

	Test Case T1.1.9.5
Coverage Item	Tcover 1.1.9.6, Tcover 1.1.9.7

Input	side = 1
	out_sig = [1,1,0,0,0]
	c_position = [352,352]
	c_state = [0,1]

State	hold_list(1) == 0 hold_list(2) == 0 hold_list(3) == -1 hold_list(4) == -1
Expected Output	left_ele.current_state == 3

	Test Case T1.1.9.6
Coverage Item	Tcover 1.1.9.6, Tcover 1.1.9.8

Input	side = 1
	out_sig = [0,1,1,0,0]
	c_position = [236,352]
	c_state = [0,1]

	Test Case T1.1.9.6
State	hold_list(1) == 0 hold_list(2) == -1 hold_list(3) == 1 hold_list(4) == -1
Expected Output	left_ele.current_state == 1

	Test Case T1.1.9.7
Coverage Item	Tcover 1.1.9.6, Tcover 1.1.9.9

Input	side = 2
	out_sig = [1,0,0,0,1]
	c_position = [226,467]
	c_state = [2,2]

State	hold_list(1) == 1 hold_list(2) == 0 hold_list(3) == -1 hold_list(4) == -1
Expected Output	left_ele.current_state == 0

	Test Case T1.1.9.8
Coverage Item	Tcover 1.1.9.6, Tcover 1.1.9.10

Input	side = 1
	out_sig = [1,1,1,0,0]
	c_position = [352,236]
	c_state = [2,2]

State	hold_list(1) == 0 hold_list(2) == 0 hold_list(3) == 1 hold_list(4) == -1
Expected Output	left_ele.current_state == 1

- Test coverage: 11/11=100%
- Test Result: 8 passed

T1.1.10 Test choose_hold()

```
function theone = choose_hold(obj,a,b,side)
    theone = -1;
    if obj.hold_list(a) == side % Tcover 1.1.10.1
        theone = b;
        return;
    elseif obj.hold_list(b) == side % Tcover 1.1.10.2
        theone = a;
        return;
    end
end
```

- Coverage Criteria: Branch Coverage
- Test case

	Test Case T1.1.10.1
Coverage Item	Tcover 1.1.10.1

Input	a = 1
	b = 2
	side = 0

State	hold_list(a) == side
Expected Output	theone == b

	Test Case T1.1.10.2
Coverage Item	Tcover 1.1.10.2

Input	a = 1
	b = 2
	side = 1

State	hold_list(b) == side
Expected Output	theone == a

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

T1.1.11 Test update_all()

```
function update_all(obj)
    if obj.sig_list(5) == 0 && obj.sig_list(6) == 0 && obj.sig_list(7) == 0 &&
obj.sig_list(8) == 0 && obj.sig_list(9) == 0 % Tcover 1.1.11.1
        obj.left_inner_p.whether_havesig = 0;
    else % Tcover 1.1.11.1
```

```
        obj.left_inner_p.whether_havesig = 1;
    end
    if obj.sig_list(10) == 0 && obj.sig_list(11) == 0 && obj.sig_list(12) == 0 &&
obj.sig_list(13) == 0 && obj.sig_list(14) == 0 % Tcover 1.1.11.2
        obj.right_inner_p.whether_havesig = 0;
    else % Tcover 1.1.11.3
        obj.right_inner_p.whether_havesig = 1;
    end

    % Update button light
    if obj.sig_list(1) == 0 % Tcover 1.1.11.4
        obj.ui.third_Button.BackgroundColor = [0.96,0.96,0.96];
    end
    if obj.sig_list(2) == 0 % Tcover 1.1.11.5
        obj.ui.second_Button_up.BackgroundColor = [0.96,0.96,0.96];
    end
    if obj.sig_list(3) == 0 % Tcover 1.1.11.6
        obj.ui.second_Button_down.BackgroundColor = [0.96,0.96,0.96];
    end
    if obj.sig_list(4) == 0 % Tcover 1.1.11.7
        obj.ui.first_Button.BackgroundColor = [0.96,0.96,0.96];
    end
    if obj.sig_list(5) == 0 % Tcover 1.1.11.8
        obj.ui.left_open.BackgroundColor = [0.96,0.96,0.96];
    end
    if obj.sig_list(6) == 0 % Tcover 1.1.11.9
        obj.ui.left_close.BackgroundColor = [0.96,0.96,0.96];
    end
    if obj.sig_list(7) == 0 % Tcover 1.1.11.10
        obj.ui.left_call_3.BackgroundColor = [0.96,0.96,0.96];
    end
    if obj.sig_list(8) == 0 % Tcover 1.1.11.11
        obj.ui.left_call_2.BackgroundColor = [0.96,0.96,0.96];
    end
    if obj.sig_list(9) == 0 % Tcover 1.1.11.12
        obj.ui.left_call_1.BackgroundColor = [0.96,0.96,0.96];
    end
    if obj.sig_list(10) == 0 % Tcover 1.1.11.13
        obj.ui.right_call_3.BackgroundColor = [0.96,0.96,0.96];
    end
    if obj.sig_list(11) == 0 % Tcover 1.1.11.14
        obj.ui.right_call_2.BackgroundColor = [0.96,0.96,0.96];
    end
    if obj.sig_list(12) == 0 % Tcover 1.1.11.15
        obj.ui.right_call_1.BackgroundColor = [0.96,0.96,0.96];
    end
    if obj.sig_list(13) == 0 % Tcover 1.1.11.16
        obj.ui.right_open.BackgroundColor = [0.96,0.96,0.96];
    end
    if obj.sig_list(14) == 0 % Tcover 1.1.11.17
        obj.ui.right_close.BackgroundColor = [0.96,0.96,0.96];
    end
```

```
% Update arrow light
if obj.left_ele.current_state == 1 % Tcover 1.1.11.18
    obj.ui.left_ele_up.BackgroundColor = [0.00,1.00,0.00];
    obj.ui.left_ele_down.BackgroundColor = [1.00,1.00,1.00];
elseif obj.left_ele.current_state == 2 % Tcover 1.1.11.19
    obj.ui.left_ele_down.BackgroundColor = [0.00,1.00,0.00];
    obj.ui.left_ele_up.BackgroundColor = [1.00,1.00,1.00];
else % Tcover 1.1.11.20
    obj.ui.left_ele_up.BackgroundColor = [1.00,1.00,1.00];
    obj.ui.left_ele_down.BackgroundColor = [1.00,1.00,1.00];
end

if obj.right_ele.current_state == 1 % Tcover 1.1.11.21
    obj.ui.right_ele_up.BackgroundColor = [0.00,1.00,0.00];
    obj.ui.right_ele_down.BackgroundColor = [1.00,1.00,1.00];
elseif obj.right_ele.current_state == 2 % Tcover 1.1.11.22
    obj.ui.right_ele_down.BackgroundColor = [0.00,1.00,0.00];
    obj.ui.right_ele_up.BackgroundColor = [1.00,1.00,1.00];
else % Tcover 1.1.11.23
    obj.ui.right_ele_up.BackgroundColor = [1.00,1.00,1.00];
    obj.ui.right_ele_down.BackgroundColor = [1.00,1.00,1.00];
end

% Update digit
% Update all state of back_end of left elevator
if obj.ui.inner_left_showcase.Position(2) == 467 % Tcover 1.1.11.24
    obj.left_ele.current_floor = 3;
    obj.ui.left_digit.Text = "3";
    obj.ui.left_third_digit.Text = "3";
    obj.ui.left_second_digit.Text = "3";
    obj.ui.left_first_digit.Text = "3";
elseif obj.ui.inner_left_showcase.Position(2) == 352 % Tcover 1.1.11.25
    obj.left_ele.current_floor = 2;
    obj.ui.left_digit.Text = "2";
    obj.ui.left_third_digit.Text = "2";
    obj.ui.left_second_digit.Text = "2";
    obj.ui.left_first_digit.Text = "2";
elseif obj.ui.inner_left_showcase.Position(2) == 236 % Tcover 1.1.11.26
    obj.left_ele.current_floor = 1;
    obj.ui.left_digit.Text = "1";
    obj.ui.left_third_digit.Text = "1";
    obj.ui.left_second_digit.Text = "1";
    obj.ui.left_first_digit.Text = "1";
end
% Update all state of back_end of right elevator
if obj.ui.inner_right_showcase.Position(2) == 467 % Tcover 1.1.11.27
    obj.right_ele.current_floor = 3;
    obj.ui.right_digit.Text = "3";
    obj.ui.right_third_digit.Text = "3";
    obj.ui.right_second_digit.Text = "3";
    obj.ui.right_first_digit.Text = "3";
```

```
      elseif obj.ui.inner_right_showcase.Position(2) == 352 % Tcover 1.1.11.28
          obj.right_ele.current_floor = 2;
          obj.ui.right_digit.Text = "2";
          obj.ui.right_third_digit.Text = "2";
          obj.ui.right_second_digit.Text = "2";
          obj.ui.right_first_digit.Text = "2";
      elseif obj.ui.inner_right_showcase.Position(2) == 236 % Tcover 1.1.11.29
          obj.right_ele.current_floor = 1;
          obj.ui.right_digit.Text = "1";
          obj.ui.right_third_digit.Text = "1";
          obj.ui.right_second_digit.Text = "1";
          obj.ui.right_first_digit.Text = "1";
      end
    end
```

- Coverage Criteria: Branch Coverage
- Test case

	Test Case T1.1.11.1
Coverage Item	Tcover 1.1.11.1, Tcover 1.1.11.3, Tcover 1.1.11.4, Tcover 1.1.11.5, Tcover 1.1.11.6, Tcover 1.1.11.7, Tcover 1.1.11.8, Tcover 1.1.11.9, Tcover 1.1.11.10, Tcover 1.1.11.5, Tcover 1.1.11.12, Tcover 1.1.11.13, Tcover 1.1.11.14, Tcover 1.1.11.15,Tcover 1.1.11.16, Tcover 1.1.11.17, Tcover 1.1.11.18, Tcover 1.1.11.21,Tcover 1.1.11.24, Tcover 1.1.11.27
Input	---------
State	hold_list(all) == 0 c_position = [236,236]
Expected Output	left_inner_p.whether_havesig == 0

	Test Case T1.1.11.2
Coverage Item	Tcover 1.1.11.2, Tcover 1.1.11.19, Tcover 1.1.11.22, Tcover 1.1.11.25, Tcover 1.1.11.28
Input	---------
State	hold_list(all) == 1 c_position = [352,352]
Expected Output	right_inner_p.whether_havesig == 1

	Test Case T1.1.11.3

	Test Case T1.1.11.3
Coverage Item	Tcover 1.1.11.2, Tcover 1.1.11.20, Tcover 1.1.11.23, Tcover 1.1.11.26, Tcover 1.1.11.29
Input	---------
State	hold_list(all) == 1 c_position = [467,467]
Expected Output	left_inner_p.whether_havesig == 0

- Test coverage: 29/29=100%
- Test Result: 3 passed

T1.2 Instruction Test

T1.2.1 Test ass_leftget()

```
function ans = ass_leftget(obj)
    ans = 0;
    for i = 1:4
        if obj.hold_list(i) == 0
            ans = 1;return;
        end
    end
end
```

- Coverage Criteria: Statement Coverage
- Test case

	Test Case T1.2.1
Coverage Item	Tcover 1.2.1
Input	---------
State	hold_list(1) == 0
Expected Output	ans == 1

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

T1.2.2 Test ass_rightget()

```
function ans = ass_rightget(obj)
    ans = 0;
    for i = 1:4
        if obj.hold_list(i) == 1
            ans = 1;return;
        end
    end
end
```

- Coverage Criteria: Statement Coverage
- Test case

	Test Case T1.2.2
Coverage Item	Tcover 1.2.2
Input	---------
State	hold_list(1) == 1
Expected Output	ans == 1

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

T1.3 Timer Test

T1.3.1 Test delete_timer()

```
function delete_timer(obj)
    stop(obj.flush);
    delete(obj.flush);
end
```

- Coverage Criteria: Statement Coverage
- Test case

	Test Case T1.3.1
Coverage Item	Tcover 1.3.1
Input	---------
State	---------
Expected Output	Timer is empty

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

T2 Integration Test

T2.1 Dispatch and De-duplication of instructions(ass_repeat)

```
% T2.1.1
function dispatch_repeat_test2_1_1(tc)
    expect_ans = [0;0;0;0;0;0;0;0;0;0;0;0;0;0];
    start(tc.console);
    tc.press(tc.view.second_Button_up);
    pause(0.2);
    tc.press(tc.view.third_Button);
    pause(0.2);
    tc.press(tc.view.first_Button);
    pause(0.2);
    tc.press(tc.view.left_call_3);
    pause(0.2);
    tc.press(tc.view.left_call_1);
    pause(0.2);
    tc.press(tc.view.right_call_2);
    pause(0.2);
    tc.press(tc.view.right_call_3);
    pause(0.2);

    while ~(all(tc.controller.sig_list(:) == 0))
        pause(0.2);
    end
    tc.verifyEqual(tc.controller.sig_list(:),expect_ans);
end
% T2.1.2
function dispatch_repeat_test2_1_2(tc)
    expect_ans = [0;0;0;0;0;0;0;0;0;0;0;0;0;0];
    start(tc.console);
    tc.press(tc.view.left_open);
    pause(0.2);
    tc.press(tc.view.right_call_3);
    pause(0.2);

    while ~(all(tc.controller.sig_list(:) == 0))
        pause(0.2);
    end
    tc.verifyEqual(tc.controller.sig_list(:),expect_ans);
end
% T2.1.3
function dispatch_repeat_test2_1_3(tc)
    expect_ans = [0;0;0;0;0;0;0;0;0;0;0;0;0;0];
    start(tc.console);
    tc.press(tc.view.left_open);
    pause(0.2);
    tc.press(tc.view.right_call_3);
    pause(0.2);
    tc.press(tc.view.second_Button_up);
    pause(0.2);

    while ~(all(tc.controller.sig_list(:) == 0))
        pause(0.2);
    end
    tc.verifyEqual(tc.controller.sig_list(:),expect_ans);
end
```

- Test case

	Test Case T2.1.1
Coverage Item	Tcover 2.1.1
Input	press second_Button_up -> press third_Button -> press first_Button -> press left_call_3 -> press left_call_1 -> press right_call_2 -> press right_call_3
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle second_Button_up and third_Button to third floor and then handle left_call_1 to the first floor; Right elevator handle right_call_2 and right_call_3 to the third floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T2.1.2
Coverage Item	Tcover 2.1.2
Input	press left_open -> press right_call_3
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s

	Test Case T2.1.2
Expected Output	Left elevator handle left_open, when open to max, wait 5s and close the door. Right elevator handle right_call_3 to the third floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T2.1.3
Coverage Item	Tcover 2.1.3
Input	press left_open -> press right_call_3 -> press second_Button_up
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle left_open, when open to max, wait 5s and close the door. Right elevator handle both second_Button_up and right_call_3 to the second floor and then to the third floor. Also all signal will be handled(all(sig_list) == 0)

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

T2.2 Inner_handle_process, priority_instr and dispatch oneside

```
% T2.2.1
function inner_handle_priority2_2_1(tc)
    expect_ans = [0;0;0;0;0;0;0;0;0;0;0;0;0;0];
    start(tc.console);
    tc.press(tc.view.left_open);
    pause(0.2);
    tc.press(tc.view.left_call_2);
    pause(0.2);
    tc.press(tc.view.left_call_3);
    pause(0.2);
    tc.press(tc.view.second_Button_down);
    pause(0.2);
    tc.press(tc.view.third_Button);
    pause(0.2);

    while ~(all(tc.controller.sig_list(:) == 0))
        pause(0.2);
    end
    tc.verifyEqual(tc.controller.sig_list(:),expect_ans);
end
% T2.2.2
function inner_handle_priority2_2_2(tc)
    expect_ans = [0;0;0;0;0;0;0;0;0;0;0;0;0;0];
    start(tc.console);
    tc.press(tc.view.left_call_1);
    pause(0.2);
    tc.press(tc.view.left_call_3);
    pause(0.2);
    tc.press(tc.view.second_Button_down);
    pause(0.2);
    tc.press(tc.view.second_Button_up);
    pause(0.2);
    tc.press(tc.view.right_call_3);
    pause(0.2);
    tc.press(tc.view.third_Button);
    pause(0.2);

    while ~(all(tc.controller.sig_list(:) == 0))
        pause(0.2);
    end
    tc.verifyEqual(tc.controller.sig_list(:),expect_ans);
end
```

- Test case

	Test Case T2.2.1
Coverage Item	Tcover 2.2.1
Input	press left_open -> press left_call_2 -> press left_call_3 -> press second_Button_down -> press third_Button
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle left_open, when open to max, wait 5s and close the door. Right elevator handle both second_Button_up and right_call_3 to the second floor and then to the third floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T2.2.2

	Test Case T2.2.2
Coverage Item	Tcover 2.2.2
Input	press left_call_1 -> press left_call_3 -> press second_Button_down -> press second_Button_up -> press right_call_3 -> press third_Button
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle left_call_1, second_Button_up, left_call_3. Right elevator handle both second_Button_down, but no downside instruction. So continue to handle right_call_3. Also all signal will be handled(all(sig_list) == 0)

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

T2.3 Look_up and elevator movement

```
% T2.3
function lookup_move_2_3(tc)
    expect_ans = [0;0;0;0;0;0;0;0;0;0;0;0;0;0];
    start(tc.console);
    tc.press(tc.view.left_call_3);
    pause(0.2);
    tc.press(tc.view.right_call_3);
    pause(0.2);

    while ~(all(tc.controller.sig_list(:) == 0))
        pause(0.2);
    end
    tc.verifyEqual(tc.controller.sig_list(:),expect_ans);
end
```

- Test case

	Test Case T2.3
Coverage Item	Tcover 2.3

	Test Case T2.3
Input	press left_call_3 -> press right_call_3
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle left_call_3 to the third floor. Right elevator handle right_call_3 to the third floor. Also all signal will be handled(all(sig_list) == 0)

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

T2.4 Additional function schedule algorithm test

```
% T2.4.1
function bonus_2_4_1(tc)
    expect_ans = [0;0;0;0;0;0;0;0;0;0;0;0;0;0];
    start(tc.console);
    tc.press(tc.view.second_Button_up);
    pause(0.2);
    tc.press(tc.view.third_Button);
    pause(0.2);

    while ~(all(tc.controller.sig_list(:) == 0))
        pause(0.2);
    end
    tc.verifyEqual(tc.controller.sig_list(:),expect_ans);
end
% T2.4.2
function bonus_2_4_2(tc)
    expect_ans = [0;0;0;0;0;0;0;0;0;0;0;0;0;0];
    start(tc.console);
    tc.press(tc.view.second_Button_down);
    pause(0.2);
    tc.press(tc.view.third_Button);
    pause(0.2);

    while ~(all(tc.controller.sig_list(:) == 0))
        pause(0.2);
    end
    tc.verifyEqual(tc.controller.sig_list(:),expect_ans);
end
% T2.4.3
function bonus_2_4_3(tc)
    expect_ans = [0;0;0;0;0;0;0;0;0;0;0;0;0;0];
    start(tc.console);
    tc.press(tc.view.first_Button);
    pause(0.2);
    tc.press(tc.view.second_Button_up);
    pause(0.2);

    while ~(all(tc.controller.sig_list(:) == 0))
        pause(0.2);
    end
    tc.verifyEqual(tc.controller.sig_list(:),expect_ans);
end
% T2.4.4
function bonus_2_4_4(tc)
    expect_ans = [0;0;0;0;0;0;0;0;0;0;0;0;0;0];
    start(tc.console);
    tc.press(tc.view.left_call_2);
    pause(0.2);
    tc.press(tc.view.third_Button);
    pause(0.2);

    while ~(all(tc.controller.sig_list(:) == 0))
        pause(0.2);
    end
    tc.verifyEqual(tc.controller.sig_list(:),expect_ans);
end
% T2.4.5
function bonus_2_4_5(tc)
    expect_ans = [0;0;0;0;0;0;0;0;0;0;0;0;0;0];
    start(tc.console);
    tc.press(tc.view.left_call_3);
    pause(0.2);
    tc.press(tc.view.third_Button);
    pause(0.2);

    while ~(all(tc.controller.sig_list(:) == 0))
        pause(0.2);
    end
    tc.verifyEqual(tc.controller.sig_list(:),expect_ans);
end
% T2.4.6
function bonus_2_4_6(tc)
    expect_ans = [0;0;0;0;0;0;0;0;0;0;0;0;0;0];
    start(tc.console);
    tc.press(tc.view.right_call_2);
    pause(0.2);
    tc.press(tc.view.third_Button);
    pause(0.2);

    while ~(all(tc.controller.sig_list(:) == 0))
        pause(0.2);
    end
    tc.verifyEqual(tc.controller.sig_list(:),expect_ans);
```

```
end
% T2.4.7
function bonus_2_4_7(tc)
    expect_ans = [0;0;0;0;0;0;0;0;0;0;0;0;0;0];
    start(tc.console);
    tc.press(tc.view.right_call_3);
    pause(0.2);
    tc.press(tc.view.third_Button);
    pause(0.2);

    while ~(all(tc.controller.sig_list(:) == 0))
        pause(0.2);
    end
    tc.verifyEqual(tc.controller.sig_list(:),expect_ans);
end
```

- Test Case

	Test Case T2.4.1
Coverage Item	Tcover 2.4.1
Input	press second_Button_up -> press third_Button
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle second_Button_up and third_Button to the second floor and then to the third floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T2.4.2
Coverage Item	Tcover 2.4.2
Input	press second_Button_down -> press third_Button
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s

	Test Case T2.4.2
Expected Output	Left elevator handle second_Button_down and third_Button to the third floor first, then to the second floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T2.4.3
Coverage Item	Tcover 2.4.3
Input	press first_Button -> press second_Button_up
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle first_Button and open directly and then second_Button_up to the second floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T2.4.4
Coverage Item	Tcover 2.4.4
Input	press left_call_2 -> press third_Button
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle left_call_2 and third_Button to the second floor and then to the third floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T2.4.5
Coverage Item	Tcover 2.4.5
Input	press left_call_3 -> press third_Button

	Test Case T2.4.5
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle left_call_3 and third_Button to the third floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T2.4.6
Coverage Item	Tcover 2.4.6
Input	press right_call_2 -> press third_Button
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Right elevator handle right_call_2 and third_Button to the second floor and then to the third floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T2.4.7
Coverage Item	Tcover 2.4.7
Input	press right_call_3 -> press third_Button
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Right elevator handle right_call_3 and third_Button to the third floor. Also all signal will be handled(all(sig_list) == 0)

- Test coverage: 7/7=100%
- Test Result: 7 passed

T3 Functional Test

T3.1 Request target direction

- Test Case

	Test Case T3.1.1
Coverage Item	Tcover 3.1.1
Input	press first_Button
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle first_Button and open directly. Also all signal will be handled(all(sig_list) == 0)

	Test Case T3.1.2
Coverage Item	Tcover 3.1.2
Input	press second_Button_down
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle second_Button_down and go to the second floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T3.1.3
Coverage Item	Tcover 3.1.3
Input	press second_Button_up
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle second_Button_up and go to the second floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T3.1.4
Coverage Item	Tcover 3.1.4

	Test Case T3.1.4
Input	press third_Button
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle third_Button and go to the third floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T3.1.5
Coverage Item	Tcover 3.1.5
Input	press second_Button_down -> press first_Button
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle first_Button and open directly. Right elevator handle second_Button_down and go to the second floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T3.1.6
Coverage Item	Tcover 3.1.6
Input	press second_Button_up -> press first_Button
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle first_Button and open directly. Right elevator handle second_Button_up and go to the second floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T3.1.7
Coverage Item	Tcover 3.1.7
Input	press third_Button -> press first_Button

	Test Case T3.1.7
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle first_Button and open directly. Right elevator handle third_Button and go to the third floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T3.1.8
Coverage Item	Tcover 3.1.8
Input	press second_Button_up -> press second_Button_down
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle second_Button_up and go to the second floor. Right elevator handle second_Button_down and go to the second floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T3.1.9
Coverage Item	Tcover 3.1.9
Input	press second_Button_down -> press third_Button
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle second_Button_down and go to the second floor. Right elevator handle third_Button and go to the third floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T3.1.10
Coverage Item	Tcover 3.1.10

	Test Case T3.1.10
Input	press second_Button_up -> press third_Button
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle second_Button_up and third_Button go to the second floor then go to the third floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T3.1.11
Coverage Item	Tcover 3.1.11
Input	press first_Button press second_Button_up -> press second_Button_down
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle first_Button and open directly, then handle second_Button_down and go to the second floor. Right elevator handle second_Button_up and go to the second floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T3.1.12
Coverage Item	Tcover 3.1.12
Input	press first_Button press third_Button -> press second_Button_down
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s

	Test Case T3.1.12
Expected Output	Left elevator handle first_Button and open directly, then handle second_Button_down and go to the second floor. Right elevator handle third_Button and go to the third floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T3.1.13
Coverage Item	Tcover 3.1.13
Input	press first_Button press second_Button_up -> press third_Button
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle first_Button and open directly. Right elevator handle second_Button_up and third_Button and go to the second and then go to the third floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T3.1.14
Coverage Item	Tcover 3.1.14
Input	press third_Button -> press second_Button_up -> press second_Button_down
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle second_Button_up and third_Button and go to the second and then go to the third floor. Right elevator handle second_Button_down go to the second floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T3.1.15

	Test Case T3.1.15
Coverage Item	Tcover 3.1.15
Input	press first_Button -> press second_Button_up -> press second_Button_down -> press third_Button
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle first_Button and open directly, then handle second_Button_down and go to the second floor. Right elevator handle second_Button_up and third_Button and go to the second and then go to the third floor. Also all signal will be handled(all(sig_list) == 0)

- Test coverage: 15/15=100%
- Test Result: 15 passed

T3.2 Request target floor

- Test Case

	Test Case T3.2.1
Coverage Item	Tcover 3.2.1
Input	press left_call_1 -> press right_call_1
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle left_call_1 and open directly. Right elevator handle right_call_1 and open directly. Also all signal will be handled(all(sig_list) == 0)

	Test Case T3.2.2
Coverage Item	Tcover 3.2.2

	Test Case T3.2.2
Input	press left_call_2 -> press right_call_2
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle left_call_2 and go to the second floor. Right elevator handle right_call_2 and go to the second floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T3.2.3
Coverage Item	Tcover 3.2.3
Input	press left_call_3 -> press right_call_3
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle left_call_3 and go to the third floor. Right elevator handle right_call_3 and go to the third floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T3.2.4
Coverage Item	Tcover 3.2.4
Input	press left_call_1 -> press right_call_1 -> press left_call_2 -> press right_call_2 -> press left_call_3 -> press right_call_3
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s

	Test Case T3.2.4
Expected Output	Left elevator handle left_call_1, left_call_2, left_call_3 and ope n directly first and then go to second floor, third floor. Right elevator handle right_call_1, right_call_2, right_call_3 and ope n directly first and then go to second floor, third floor. Also all signal will be handled(all(sig_list) == 0)

	Test Case T3.2.5
Coverage Item	Tcover 3.2.5
Input	press left_call_1 -> press second_Button_down -> press second_Button_up -> press right_call_1 -> press right_call_3
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle left_call_1 and open directly then handle second_Button_up and go to the second floor. Right elevator handle second_Button_down and go to the second floor, then handle right_call_1 and go to the first floor, finally handle right_call_3 and go to the third floor. Also all signal will be handled(all(sig_list) == 0)

- Test coverage: 5/5=100%
- Test Result: 5 passed

T3.3 Open the door

- Test Case

	Test Case T3.3.1
Coverage Item	Tcover 3.3.1
Input	press left_open -> press right_open

	Test Case T3.3.1
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle left_open and open directly. Right elevator handle right_open and open directly. Also all signal will be handled(all(sig_list) == 0)

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

T3.4 Close the door

- Test Case

	Test Case T3.4.1
Coverage Item	Tcover 3.4.1
Input	press left_open -> press right_open -> press left_close -> press right_close
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle left_call_1 and open directly, when open to the maximum, call left_close and close the door without waiting. Right elevator handle right_call_1 and open directly, when open to the maximum, call right_close and close the door without waiting. Also all signal will be handled(all(sig_list) == 0)

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

T3.5 Cancel the instruction

- Test Case

	Test Case T3.5.1

	Test Case T3.5.1
Coverage Item	Tcover 3.5.1
Input	press left_call_1 -> press left_call_2 -> press left_call_3 -> press left_call_3 -> press left_call_3 -> press left_call_2 -> press left_call_2
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle left_call_1 and open directly, when open to the maximum, call left_call_3 twice and cancel this instruction, also call left_call_2 twice and cancel this instruction. Also all signal will be handled(all(sig_list) == 0)

	Test Case T3.5.2
Coverage Item	Tcover 3.5.2
Input	press third_Button -> press left_call_2 -> press left_call_1 -> press left_call_1 -> press left_call_1
State	Up/Down period == 0.1s initial a/v == 0 open/close period == 0.1s open_Max == 5s
Expected Output	Left elevator handle left_call_2 and go to the second floor, when open to the maximum, call left_call_1 twice and cancel this instruction, then handle third_Button and go to the third floor. Also all signal will be handled(all(sig_list) == 0)

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

T4 Risk Management

M1. User press open/close when moving

Risk analysis

- Hazard Situation : The elevator open/close during moving action
- Possible Cause :
-
  1. User press open when moving
  2. User press close when moving

Risk evaluation

For a system , it is catastrophic.

Risk control

If the elevator is moving, we eliminate the open/close press signal.

M2. Elevator moved before the door was closed

Risk analysis

- Hazard Situation : The elevator moved before the door was closed
- Possible Cause :
-
  1. User press inner call before door closed
  2. User press outer panel before door closed

Risk evaluation

For a system , it is catastrophic.

Risk control

If the door wasn't closed, the elevator's state isn't "stop" actually. Our implement limit it can move only state is "stop" so that avoid this risk.

M3. Elevator change unfinished instruction

Risk analysis

- Hazard Situation : The elevator change unfinished instruction without holding current command
- Possible Cause :
-
  1. User press inner call when executing an instruction.
  2. User press outer panel when executing an instruction.

Risk evaluation

For a system , it is catastrophic.

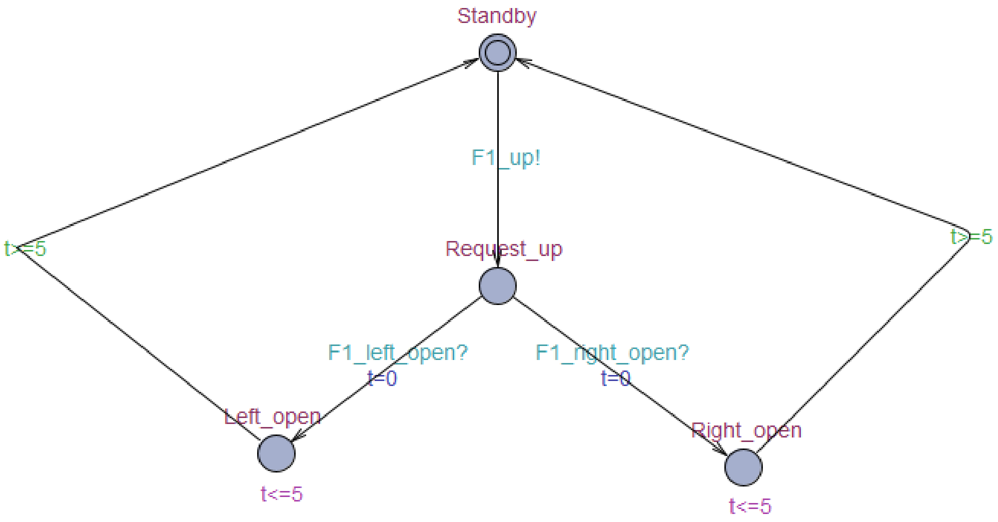
Risk control

We design a complete schedule algorithm that if one instruction has been hold by an elevator, it always execute it with high priority so that we can avoid this risk.

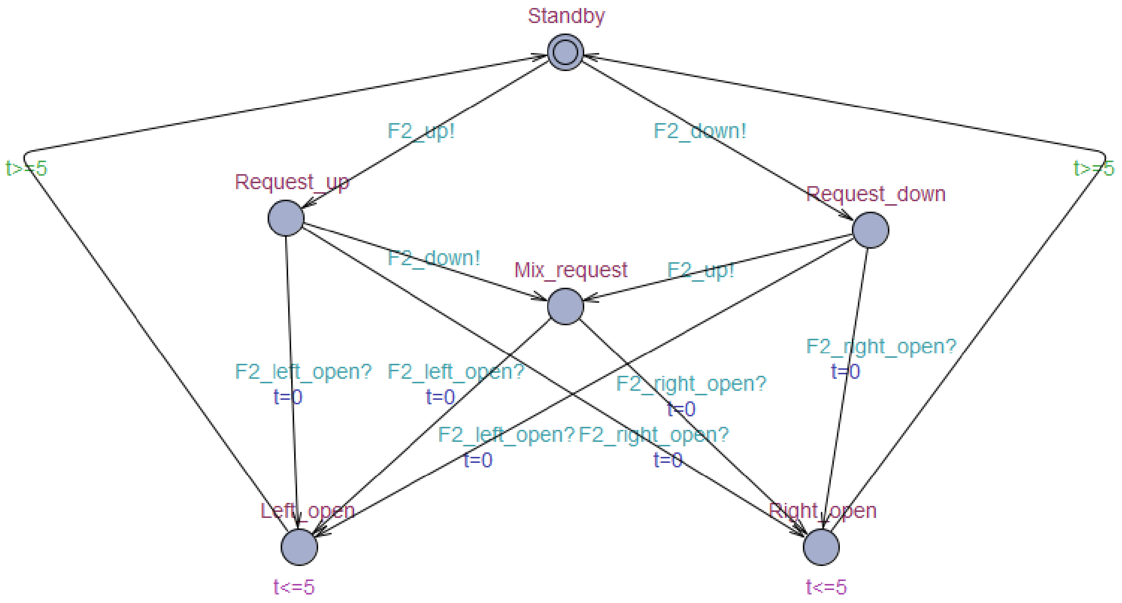
T5 Uppaal Model Test

T5.1 Model

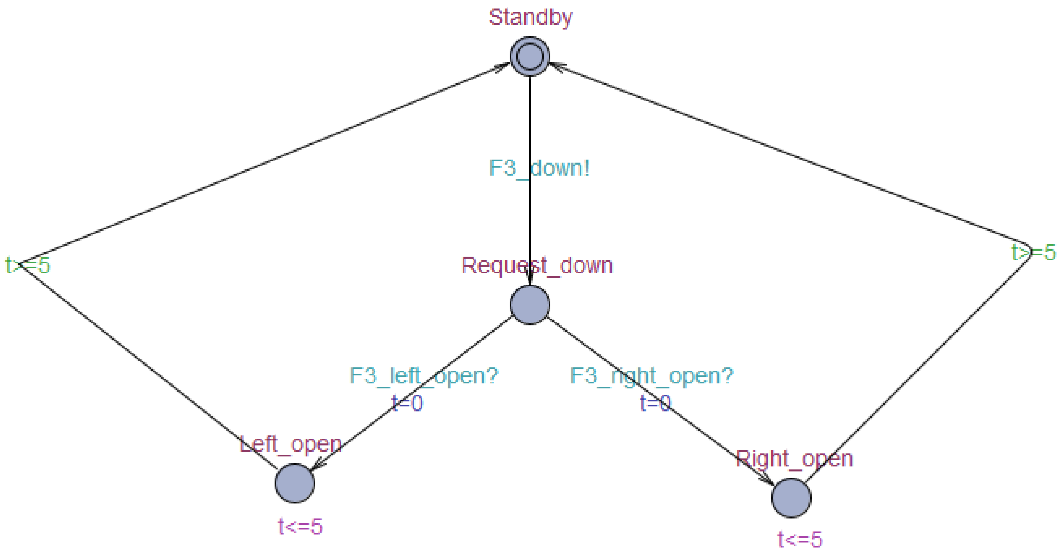
- First floor model



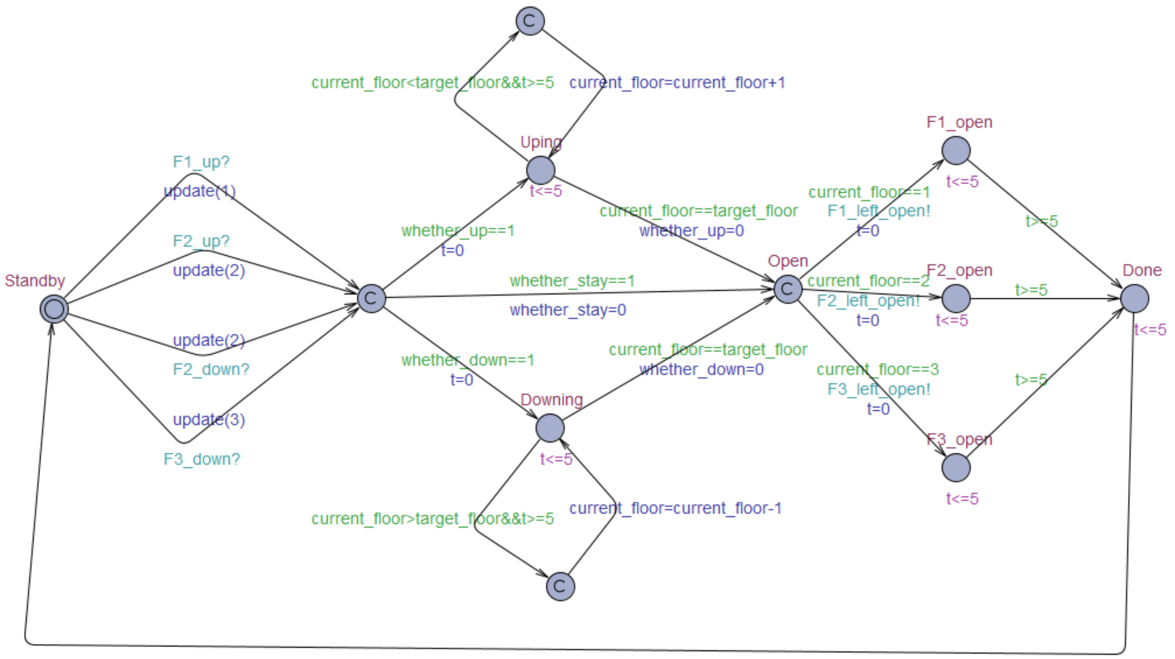
- Second floor model



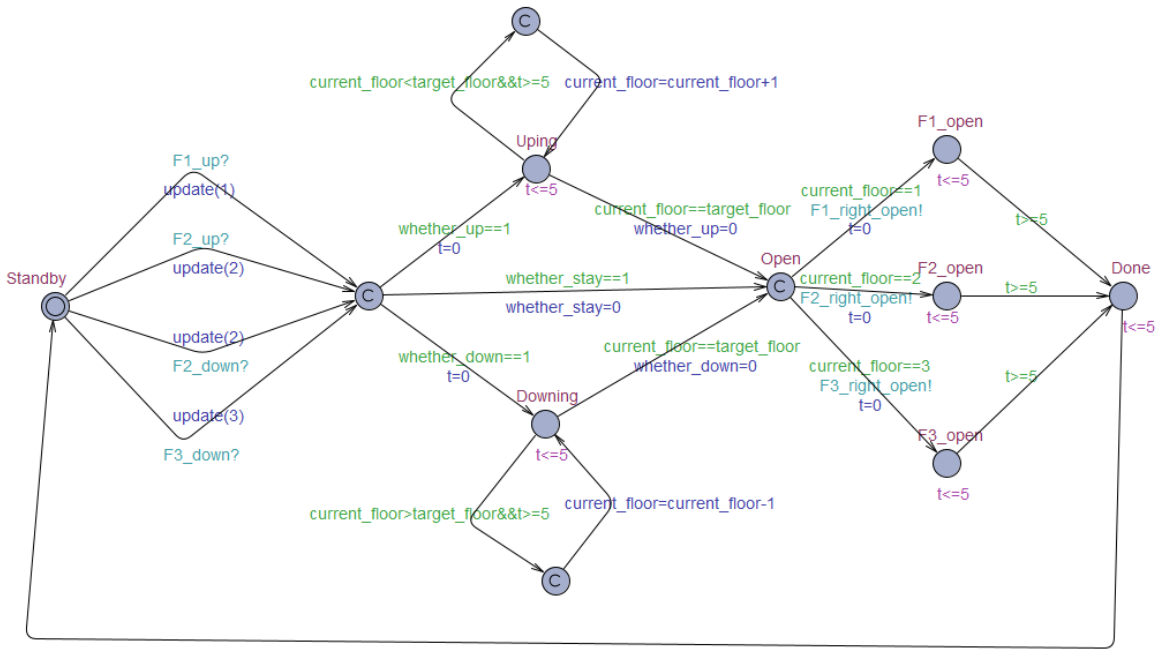
- Third floor model



- Left_elevator model



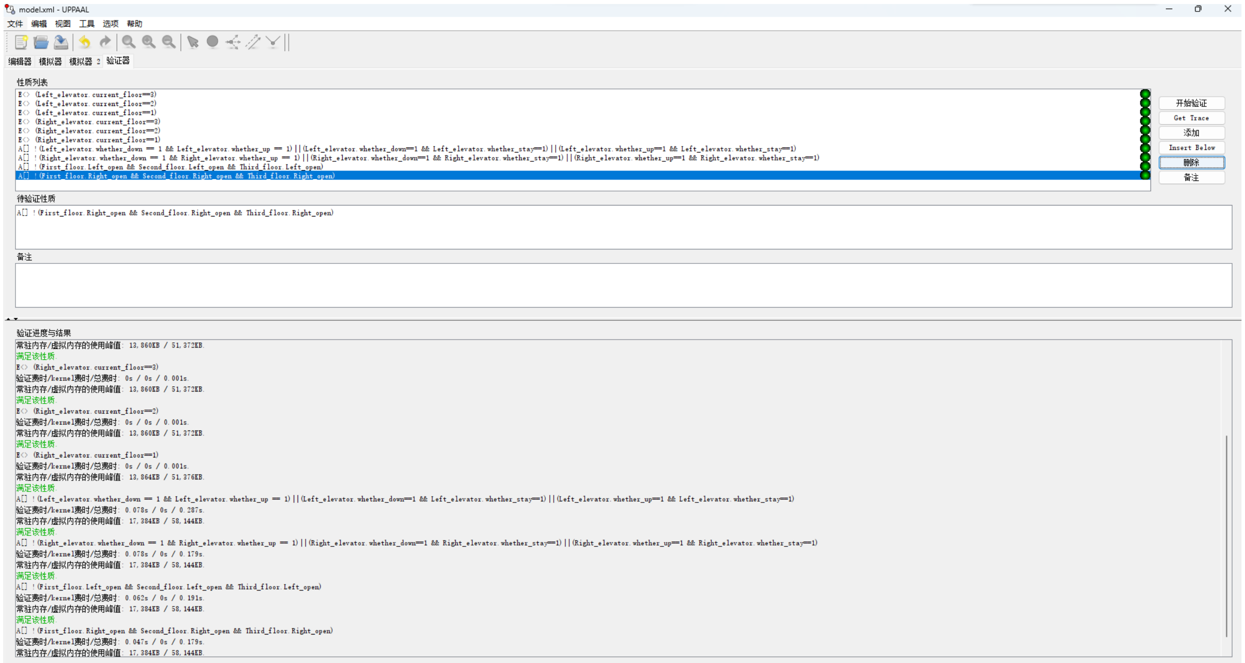
- Right_elevator model



T5.2 Property Test

1. Left_elevator can arrive floor 3
2. Left_elevator can arrive floor 2
3. Left_elevator can arrive floor 1
4. Right_elevator can arrive floor 3
5. Right_elevator can arrive floor 2
6. Right_elevator can arrive floor 1
7. When Left_elevator is in mode "up", it cannot be in "standby" or "down".

8. When Left_elevator is in mode "down", it cannot be in "standby" or "up".
9. When Left_elevator is in mode "standby", it cannot be in "up" or "down".
10. When Right_elevator is in mode "up", it cannot be in "standby" or "down".
11. When Right_elevator is in mode "down", it cannot be in "standby" or "up".
12. When Right_elevator is in mode "standby", it cannot be in "up" or "down".
13. When Floor 1 left door opens, other floors' left floor cannot open.
14. When Floor 2 left door opens, other floors' left floor cannot open.
15. When Floor 3 left door opens, other floors' left floor cannot open.
16. When Floor 1 right door opens, other floors' left floor cannot open.
17. When Floor 2 right door opens, other floors' left floor cannot open.
18. When Floor 3 right door opens, other floors' left floor cannot open.



性质列表

```
E<> (Left_elevator.current_floor==3)
E<> (Left_elevator.current_floor==2)
E<> (Left_elevator.current_floor==1)
E<> (Right_elevator.current_floor==3)
E<> (Right_elevator.current_floor==2)
E<> (Right_elevator.current_floor==1)
A[] !(Left_elevator.whether_down == 1 && Left_elevator.whether_up == 1)||(Left_elevator.whether_down==1 && Left_elevator.whether_stay==1)||(Left_elevator.whether_up==1 && Left_elevator.whether_stay==1)
A[] !(Right_elevator.whether_down == 1 && Right_elevator.whether_up == 1)||(Right_elevator.whether_down==1 && Right_elevator.whether_stay==1)||(Right_elevator.whether_up==1 && Right_elevator.whether_stay==1)
A[] !(First_floor.Left_open && Second_floor.Left_open && Third_floor.Left_open)
A[] !(First_floor.Right_open && Second_floor.Right_open && Third_floor.Right_open)
```

待验证性质

```
A[] !(First_floor.Right_open && Second_floor.Right_open && Third_floor.Right_open)
```

验证进度与结果

```
常驻内存/虚拟内存的使用峰值: 13,860KB / 51,372KB.
满足该性质.
E<> (Right_elevator.current_floor==3)
验证费时/kernel费时/总费时: 0s / 0s / 0.001s.
常驻内存/虚拟内存的使用峰值: 13,860KB / 51,372KB.
满足该性质.
E<> (Right_elevator.current_floor==2)
验证费时/kernel费时/总费时: 0s / 0s / 0.001s.
常驻内存/虚拟内存的使用峰值: 13,860KB / 51,372KB.
满足该性质.
E<> (Right_elevator.current_floor==1)
验证费时/kernel费时/总费时: 0s / 0s / 0.001s.
常驻内存/虚拟内存的使用峰值: 13,864KB / 51,376KB.
满足该性质.
A[] !(Left_elevator.whether_down == 1 && Left_elevator.whether_up == 1)||(Left_elevator.whether_down==1 && Left_elevator.whether_stay==1)||(Left_elevator.whether_up==1 && Left_elevator.whether_stay==1)
验证费时/kernel费时/总费时: 0.078s / 0s / 0.287s.
常驻内存/虚拟内存的使用峰值: 17,384KB / 58,144KB.
满足该性质.
A[] !(Right_elevator.whether_down == 1 && Right_elevator.whether_up == 1)||(Right_elevator.whether_down==1 && Right_elevator.whether_stay==1)||(Right_elevator.whether_up==1 && Right_elevator.whether_stay==1)
验证费时/kernel费时/总费时: 0.078s / 0s / 0.179s.
常驻内存/虚拟内存的使用峰值: 17,384KB / 58,144KB.
满足该性质.
A[] !(First_floor.Left_open && Second_floor.Left_open && Third_floor.Left_open)
验证费时/kernel费时/总费时: 0.062s / 0s / 0.191s.
常驻内存/虚拟内存的使用峰值: 17,384KB / 58,144KB.
满足该性质.
A[] !(First_floor.Right_open && Second_floor.Right_open && Third_floor.Right_open)
验证费时/kernel费时/总费时: 0.047s / 0s / 0.179s.
常驻内存/虚拟内存的使用峰值: 17,384KB / 58,144KB.
```