Elevator Control

ITI8531 Software Synthesis and Verification

Mike Camara

Introduction

Following a set of requirements, I have created an elevator control system with an elevator manager orchestrating the different operational and functional components except for the user interactions. For simplicity, we restrict our analysis to 4 floors.

Functional Requirements

1. ✅ The Elevator Manager has the following elevator states {inService, floorStop, noOperations, inMaintenance}.
2. ✅ When a elevatorCall or floorCall is requested it should be registered in the service queue.
3. ✅ Impose priority to call requests.
4. ✅ When the call is received, assign a direction (forward or reverse) and service the call in the queue.
5. ✅ The lift should not operate if the user selects the same floor.
6. ✅ Execute the task queue and provide the service to the floor.
7. ✅ The motor control should latch the Direction and Enable the operation.
8. ✅ The sensor inputs should cut out the queue when the elevator reaches the floor and load the queue with a new value. It should service the requests coming in descending and ascending order while moving up and down, i.e., if there is a request from floor 2 after floor 3, still floor 2 service requests can be handled while going up.
9. ✅ The above task should be repeated for Elevator Control command as well.
10. ✅ PickUp and dropOff operations, de-register the current floor from the queue, set the operating mode to be Idle and doorOpen status to be 1.
11. ✅ Update the elevator status after this operation and continue with the service queue.

Safety Requirements

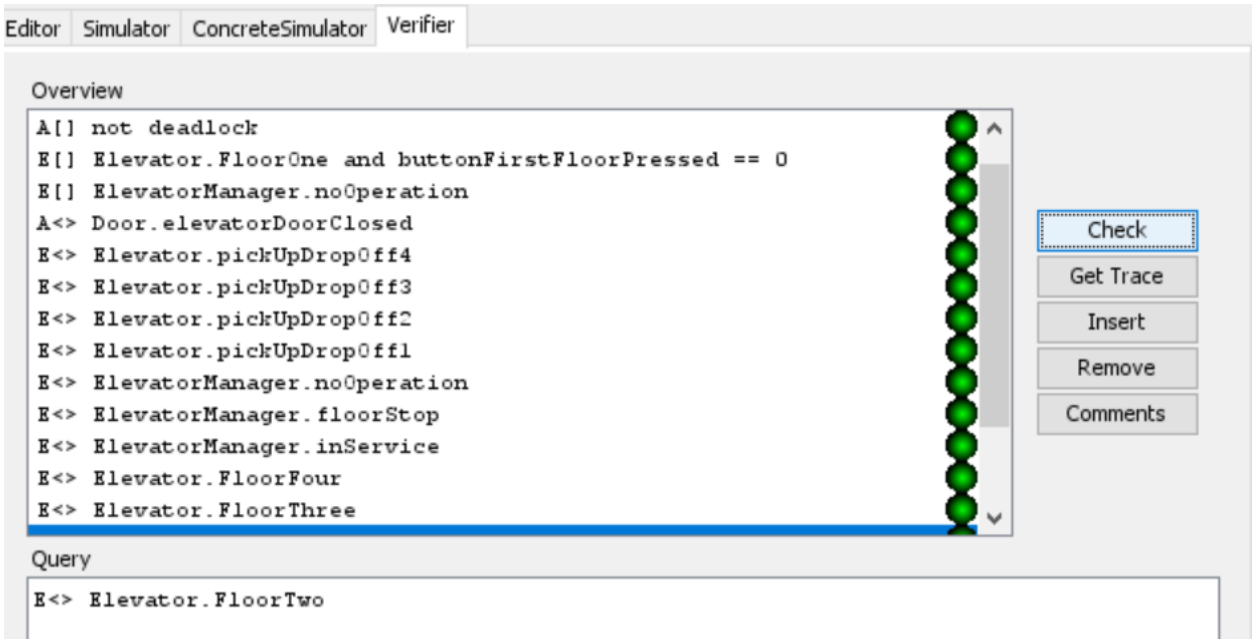
1. ✅ During dropping and picking operations keep the elevator stopped.
2. ✅ For servicing any floor control, the elevator door needs to be closed, the floor button is depressed, no overload or any other stopping operations initiated. Once the door opens the status of the lift should be brought to floor-stop.
3. ✅ Door closing is achieved after this either by depressing the door close button or when the user selects the floor (a timing will be included later here) and there are no warnings of users coming in from a motion detector sensor.

Comfort Requirements

1. ✅ Floor panel should have the following up and down button is depressed by the user, the up or down LED should glow. They can both glow together and the complimentary operations should be cut-out on reaching the floor.
2. ✅ In the elevatorPanel the floorNumber should be displayed.
3. ✅ Update the car-display on the floor when a button is depressed.
4. ✅ Based on the assigned direction show the Up and Down display in the elevatorDisplay and floorDisplay.

Verification

I used the Verification Server to simulate the internal behavior of components (states, transitions, transition conditions, and updates) and verify them against the requirements.



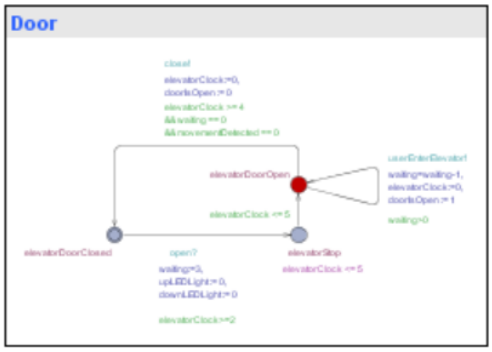
Architecture design

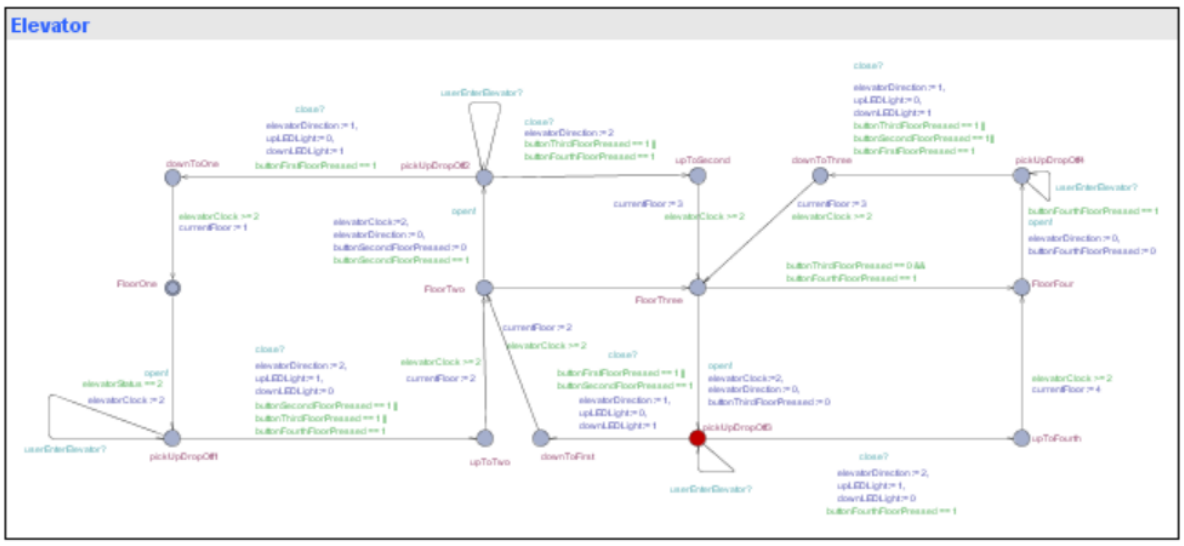
First I extracted the components from requirements and then I distinguished them into active and passive components.

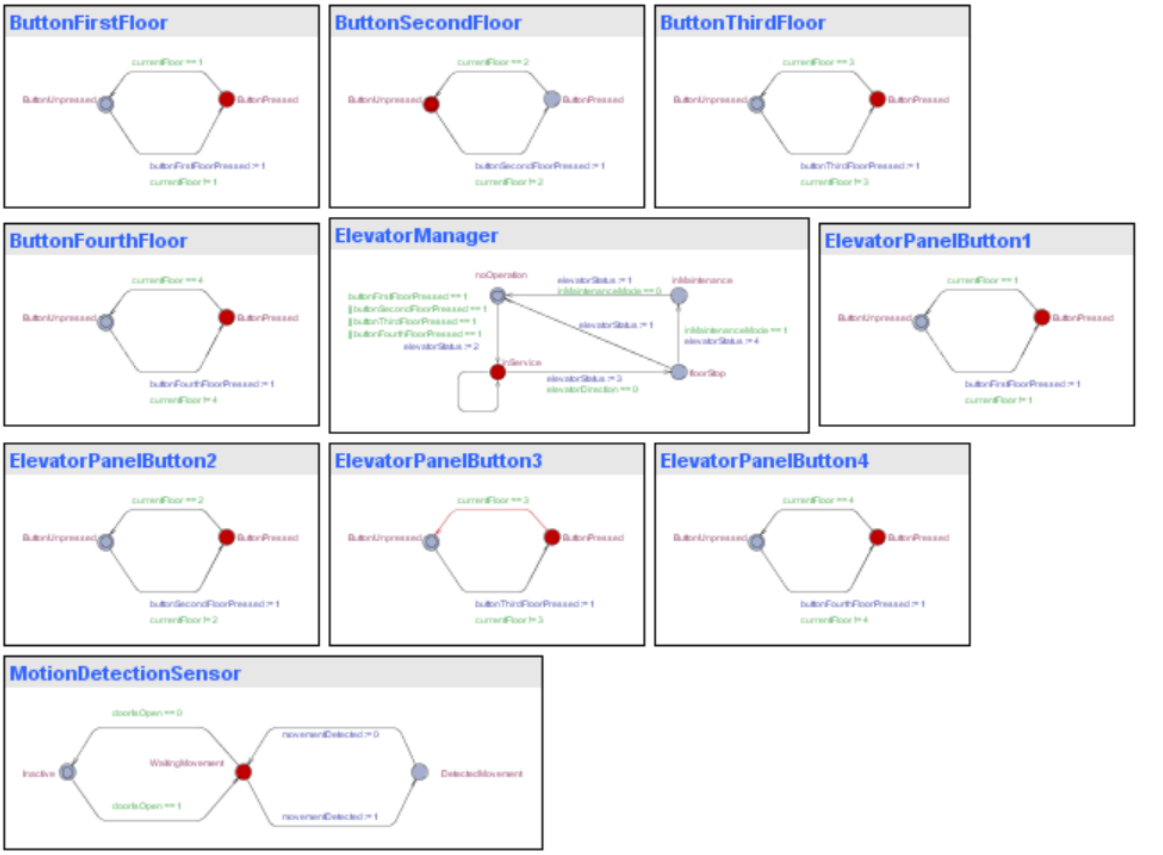
The active components such as Elevator, Door, Buttons, and Elevator Managers were then assigned to their own behavior.

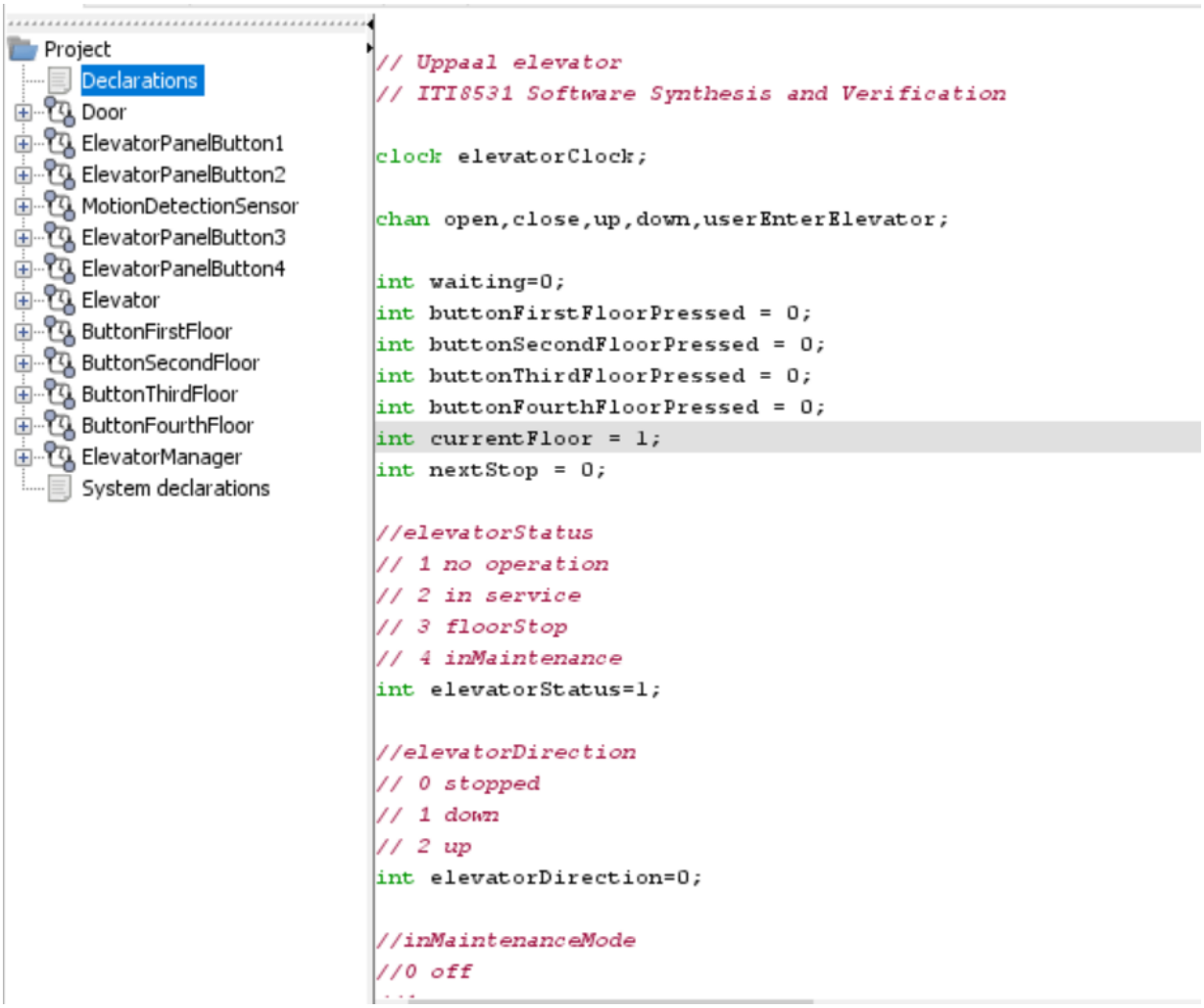
Using the UPPAAL g-raphical user interface I have specified the internal behavior of components and interfaces including the synchronization and data links between components and adding timing specification to the behavior (timed automata).

System Screenshots






ButtonFirstFloor
ButtonSecondFloor
ButtonThirdFloor
ButtonFourthFloor
ElevatorManager
ElevatorPanelButton1
ElevatorPanelButton2
ElevatorPanelButton3
ElevatorPanelButton4
MotionDetectionSensor
Inactive
WaitingMovement
DetectedMovement


Project
Declarations
Door
ElevatorPanelButton1
ElevatorPanelButton2
MotionDetectionSensor
ElevatorPanelButton3
ElevatorPanelButton4
Elevator
ButtonFirstFloor
ButtonSecondFloor
ButtonThirdFloor
ButtonFourthFloor
ElevatorManager
System declarations
// Uppaal elevator
// ITI8531 Software Synthesis and Verification
clock elevatorClock;
chan open,close,up,down,userEnterElevator;
int waiting=0;
int buttonFirstFloorPressed = 0;
int buttonSecondFloorPressed = 0;
int buttonThirdFloorPressed = 0;
int buttonFourthFloorPressed = 0;
int currentFloor = 1;
int nextStop = 0;
//elevatorStatus
// 1 no operation
// 2 in service
// 3 floorStop
// 4 inMaintenance
int elevatorStatus=1;
//elevatorDirection
// 0 stopped
// 1 down
// 2 up
int elevatorDirection=0;
//inMaintenanceMode
//0 off