Elevator Requirements

18-540 Distributed Embedded Systems Project
Fall 2000

Updated September 17, 2000

Change log:
9/4 - significantly updated since draft of September 3 to fix a variety of bugs
9/8 – minor change to Drive object; DoorMotor updated to include requirement for updating physical door position model.
9/9 – fixed a problem in 1.5.2 (for the second time – hopefully MS Word lets it stick this time)
9/13 – The dispatcher is allowed to set DesiredDwell.
9/13 – DoorClosed, HallCall and CarCall are allowable inputs to dispatcher.
9/13 – DoorOpen[j] and DesiredFloor are allowable inputs to CarPositionControl.
9/14 – clarified that top and bottom floors each have only one hall call button.
9/14 – made the passengers explicitly activate door reversal when appropriate.
9/14 – changed "Hall_Lantern" to "CarLantern" (notational correction)
9/14 – change from Drive to Drivespeed in 6.5
9/15 – changed constraint 10.3 in an attempt to make it more clear
9/17 – added HoistwayLimit[d] to the input list and used drivespeed in drive control
9/21 – fixed CarLight /CarCall in #9 to reflect the lack of directional association
9/28 – Fixed 11.6 again. Word hates me.

Elevator Top-Level Requirements

- All passengers shall eventually be delivered to their intended destination floor.
- Any unsafe condition shall cause an emergency stop.
- An emergency stop should never occur.
- Performance shall be optimized to the extent possible, where performance is defined by the formula:
  (4 * average_passenger_delivery_time) + maximum_passenger_delivery_time
  Performance is improved by reducing that value (short delivery times are better). Delivery time is counted from the time a passenger arrives at a floor to begin a trip and ends when that passenger exits the elevator car. (Note: this is an arbitrary formula for this project, but the general idea holds true for real elevators.)

Note:

The full set of example requirements provided should result in an elevator with safe behavior that meets top-level requirements other than having poor optimization of performance. While it is obvious that the Dispatcher behavior can be optimized, there are also other, more subtle, behavioral optimizations possible as well.

Notation:

The Building the elevator is in has floors numbered from 1 .. MaxFloor.

Floor #1 is the Lobby.

"[...]" indicates an array of objects or values. There are f*d separate and distinct AtFloor[f,d] sensors – d of them at each floor f.
"(...)" indicates a list of values associated with a sensor/actuator. Single-valued inputs/outputs can have the "(…)" omitted as a notational convenience.

The suffices "_up" "_down" and "_stop" may be used in lieu of a direction subscript. e.g.:

HallCall_up[f]=HallCall[f,up]
HallCall_down[f]=HallCall[f,down]
AtFloor_stop[f]=AtFloor[f,stop]

(Note that in the above examples, "f" is a floor number.)

Similarly, subscripting may be eliminated in general if desired using an underscore notation. e.g.:

CarLantern_down
EmergencyBrake_Engage

Multi-attributed items may have a particular state referred to by concatenating elements, e.g.: Motor might have state FastUp or SlowDown. ("StopStop" can always be abbreviated by "Stop")

Single attributes of a multi-attribute item are referred to using "." notation. For example DesiredFloor.f refers to attribute "f" of "DesiredFloor".

If a letter instead of a value is used in a subscript, it is assumed that it can take any value. If the same letter is used in multiple clauses within a single requirement element, it is assumed that the letter takes the same value in all instances. For example, in the phrase:

"AtFloor[f,d] ... CarCall[f] ... HallCall[g,d]"

The Floor f for AtFloor and CarCall can be any valid floor, but would have to be the same floor. Similarly, direction d for AtFloor and HallCall could be any direction, but would have to be the same direction. However, the floor for HallCall might or might not be the same floor as for AtFloor and CarCall since it is a different symbolic letter.

Car refers to the elevator Car that travels in a hoistway. The movement of the Car itself is hidden within the environment model and thus only indirectly observable/controllable by the control system via sensors and actuators.

Because ultimately we will do this all in simulation, we're going to make your life easy by telling you the initial state of the system (the "initialization" state) such as putting the elevator car at the lobby floor. In a real elevator the controllers have to figure out the system state for themselves when power is applied.

System Sensors:

These sensor values are available for use by the control system. The below-listed values will correspond to network messages in the implementation phase.

- **AtFloor[f,d](v)**: Floor proximity sensor. v={True, False}.
  One set of three per Floor[f], d={Up, Stop, Down}
  d=Stop: Indicates True at a point where the Car is approximately level with Floor[f]. It is assumed that the width of the Stop zone is such that the Drive has enough time to switch from going at Slow speed to Stop and still have the car level with the floor.
  d=Down: Above-floor position sensor. Indicates True when the Car is above Floor[f] but close enough that the Car should be traveling at slow speed to be able to stop level with Floor[f]. (In other words, this can be thought of as a "slow down " suggestion for worst case downward velocity to stop a Floor[f].)
  d=Up: the same as for d=Down, except it is the below-floor sensor, and applies to the mirror situation, when the car is traveling upward.
  Set to False at initialization, except the lobby d=Stop switch is set to True at initialization.
- **CarCall[f](v)**: Car Call buttons. v={True, False}.
  One per Floor[f], all located in the CAR.
  Set to False at initialization.
- **DoorClosed[j](v)**: Door Closed switches. v={True, False}.
  One per Door[j] for j={Left, Right}.
  Indicates True when the Door is fully closed.
  Set to True at initialization.
- **DoorOpen[j] (v)**: Door Open switches. v={True, False}.
  One per Door[j] for j={Left, Right}.
  Indicates True when the Door[j] is fully open.
  Set to False at initialization.
- **DoorReversal[j](v)**: Door Reversal sensors. v={True, False}.
  One per Door[j] for j={Left, Right}.
  Indicates True whenever the Door[j] senses an obstruction in the doorway.
  Set to False at initialization.
- **HallCall[f,d](b)**: Hall Call buttons. b={Pressed, Idle}.
  One pair per Floor[f], d={Up, Down}, located in the hallway on each Floor (topmost floor does not have an Up button; bottom floor does not have a Down button).
  Set to False at initialization.
- **HoistwayLimit[d](v)**: Safety limit switches in the hoistway. v={True, False}.
  One pair per Car, d={Up, Down}.
  A HoistwayLimit[d] switch activates when the car has over-run the hoistway limits (used as a trigger for emergency stopping). The d=Up switch is at top of hoistway; d=Down switch is at bottom of hoistway.

Set to False at initialization.

- **DriveSpeed(s,d)**: main drive speed readout. *s* is speed s={Fast, Slow, Go, Stop}
  d is direction d={Up, Down, Stop}
  One per Car. Provides information about the current drive speed set by Drive(s,d) – but this is the actual drive status rather than the status commanded by Drive(s,d). (Note that there will be a time delay between commanding the drive to change speed and the drive actually attaining that speed. DriveSpeed lets you know when the commanded speed is actually attained.)
  s=Fast whenever drive is moving faster than it would at Slow speed
  s=Slow whenever the drive is moving at Slow speed
  s=Go whenever the drive is moving, but not moving fast enough to be at Slow speed
  s=Stop whenever the drive is fully stopped.

Environmental-Only Sensors:

These are pseudo-sensor values created to explain behavior of objects external to the control system. They are not accessible to the control system, but have been used as the specifications for building the simulation system.

- **DoorPosition[j](x)**: Amount that door is open.   x={float 0 .. 50}
  One per Door[j] for j={Left, Right}.
  Value is the amount the door is open as a percentage of doorway width. Since there are two doors in the doorway, each DoorPosition can range from 0 to 50. With both doors open at 50 the entire doorway (100%) is opened overall.
  Set to 0 at initialization (door closed).
- **CarPosition(x)**: Vertical position of car.   x={float 0.. }
  Tracks the position of the car in meters (not the same as floor number).
  Set Lobby position at initialization.

System Actuators:

The below-listed values will correspond to network messages in the implementation phase. All actuators are assumed to "remember" their last commanded value and stay there unless commanded otherwise or forced otherwise by system/environment constraints.

- **DoorMotor[j](m)**: Door motor. m = {Open, Close, Stop}
  One per Car Door[j] (note that there are two Doors per Car).
  Opens and closes the door. It is permissible to transition directly from Open to Close and Close to Open without first commanding a Stop.
  Set to Stop at initialization; see DoorMotor object description for details.

- **CarLantern[d](k)**: Car Lanterns. k = {On, Off}.
  One set per Car, d={Up, Down}. These are the Up/Down arrows placed on the car doorframe. Used by Passengers on a Floor to figure out whether to enter the Car.
  Set to Off at initialization.

- **CarLight[f](k)**: Car Call Button lights. k = {On, Off}.
  One per CarCall[f] button. The light inside the car call button, used to indicate to passengers that a car call has been registered by the dispatcher.
  Set to Off at initialization.

- **CarPositionIndicator(f)**: Position Indicator in Car. f={integer 1..MaxFloor}.
  One per Car. Displays floor status information to the passengers in the Car.
  Set to 1 at initialization.

- **HallLight[f,d](k)**: Hall Call Button lights. k = {On, Off}.
  One per HallCall[f,d] button. The light inside the hall call button, used to indicate to passengers that a hall call at that Floor f has been registered by the Dispatcher for direction d.
  Set to Off at initialization.

- **Drive(s,d)**: 2-speed main elevator drive. *s* is speed s={Fast, Slow, Stop}
  d is direction d={Up, Down, Stop}
  One per Car. Moves the Car up and down the hoistway according to a velocity profile that depends on a variety of physical factors.
  Set to (Stop, Stop) at initialization; see Drive object for details.
  Note that current Drive speed can be determined via DriveSpeed(s,d)

Environmental-Only Actuators:

- **EmergencyBrake(b)**: Emergency stop brake. b={On, Off}
  Supplies emergency braking in case of safety violation such as hoistway limit over-run or movement with doors open. One per Car. Can be used exactly one time, after which elevator hoistway requires significant repair maintenance. Triggering the EmergencyBrake in simulation means that either a safety-critical sensor/actuator has been broken or your elevator controller has attempted unsafe operation. (If the EmergencyBrake activates during your final project demo due to an attempt of unsafe operation, there will be a scoring penalty.)
  Set to Off at initialization.

Control System State

The below-listed values will correspond to network messages in the implementation phase.

- **DesiredFloor(f,d)**: Dispatcher’s desired stopping Floor.
  f is desired Floor number, an integer
  d is direction d={Up, Down, Stop}
  The dispatcher uses this to indicate the next floor to stop at. A direction of Stop means that there is no preferred direction. Directions of Up and Down have the implication that the elevator is “going up” or “going down” respectively.
  This value may change dynamically and non-monotonically. Once Doors begin opening the elevator is committed to perform a full Door cycle operation and DesiredFloor can change to indicate the next Floor beyond the Floor where the Car is currently positioned.
- **DesiredDwell(n)**: Dispatcher’s desired dwell time for current Door open cycle.
  n is a long integer number of msec.
  This is an optional way for the Dispatcher to override any dwell time used by the DoorController.

Environmental Objects

These objects exist in the simulator, but are only accessible to the control system indirectly via manipulation of actuators. (These are partial specifications; internal book-keeping such as keeping track of where a passenger is and knowing the weight of the car based on passenger count to model acceleration is not visible to the simulation.)

The objects associated with the following sensors and actuators are considered simple and are not described in detail beyond definitions provided in the preceding system sensor/actuator lists. In each case interfaces consist solely of the relevant sensor/actuator state and any obvious interactions with the environment.
AtFloor[f,d]
CarCall[f]
DoorClosed[j]
DoorOpen[j]
DoorReversal[j]
HallCall[f,d]
HoistwayLimit[d]
DoorPosition[j]
CarLantern[d]
CarLight[f]
CarPositionIndicator
HallLight[f,d]
EmergencyBrake
CarPosition

Complex sensor, actuator, and environmental objects are discussed in the following sections.

1. Passenger[p] (environmental object)

Replication:

- N passengers per system; N is unlimited subject to being less than steady-state carrying capacity of elevator.

Instantiation:

- Zero passengers at initialization.
- Passengers arrive at floor landings as a Poisson process with mean interarrival times varying per floor. Lobby arrivals comprise 25% to 50% of all arrivals.
- Passengers can be on a particular Floor[m] or on the Car.

Input Interface:

- DoorPosition[j]
- CarLantern[d]
- CarLight[f]
- CarPositionIndicator
- HallLight[f,d]

Output Interface:

- CarCall[f]
- DoorReversal[j]
- HallCall[f,d]

State:

- **P_START[p]**: a constant starting floor for Passenger p.
- **P_DEST[p]**: a constant destination floor for Passenger p.
- **P_DIR[p]**: a travel direction for Passenger p, which corresponds to the direction of P_DEST[p] compared P_START[p].
- Passenger p is enqueued in an entry/exit queue, with one such queue per direction per floor, and one queue for each destination in the Car. (*i.e.*, there is a queue for going up at each floor, a separate queue for going down at each floor, and a queue for each floor for exiting the Car). The queue determines order of entry/exit and queue order is FIFO based on order of arrival of the passenger to the queue. Any passenger not at the head of a queue is blocked and must wait to be at the head of the queue before entering or exiting the Car. Obviously passengers can only exit the queue when the Car is at the correct floor going in the correct direction and the doors are sufficiently open.

CONSTRAINTS:

1.1 Passengers shall not enter a Car already containing 10 or more passengers.

1.2 Passengers are prevented from entering or exiting the Car whenever the Doors are not far enough opened.

1.2.1 The value constituting the minimum acceptable total opening distance ranges from 20% to 45% open, with the percentage randomly selected for each Passenger.

BEHAVIORS:

1.3 A Passenger p at Floor f where HallLight[f,P_DIR[p]] is Off shall press HallCall[f,P_DIR[p]].

1.3.1 Time to complete this behavior is stochastic, ranging from 400 msec to 5000 msec.

1.3.2 Each Passenger p shall press HallCall[f,P_DIR[p]] between 1 and 5 times inclusive (stochastic) while the appropriate HallLight[f,P_DIR[p]] is Off. Each time that appropriate HallLight transitions from On to Off, an additional 1-5 maximum presses per passenger is started in a similar manner.

1.4 A Passenger p at Floor f where CarLantern[f,P_DIR[p]] is On shall attempt to enter the Car if unblocked and if the Door is sufficiently far open.

1.4.1 Additionally, a Passenger p at Floor f where all CarLantern[f, *]s are Off shall attempt to enter the Car if unblocked and if the Door is sufficiently far open.

1.4.2 A passenger shall take 1 to 3 seconds (stochastic) to enter a Car once unblocked and the Door is sufficiently far open to permit entry.

1.5 A Passenger p in Car where Car_Light[P_DEST[p]] is Off shall press Car_Call[P_DEST[p]].

1.5.1 Time to complete this behavior is stochastic, ranging from 400 msec to 5000 msec.

1.5.2 Each Passenger p shall press CarCall[P_DEST[p]] between 1 and 5 times inclusive (stochastic) while the appropriate CarLight is Off. Each time the appropriate CarLight transitions from On to Off, an additional 1-5 maximum presses per passenger is started in a similar manner.

1.6 A Passenger p in the Car at where Car_Indicator[P_DEST[p]] is On shall attempt to exit the Car if unblocked and if the Door is sufficiently far open.

1.6.1 A passenger shall take 1 to 3 seconds (stochastic) to exit a Car once unblocked and the Door is sufficiently far open to permit exit.

1.7 A passenger entering the Car shall wait until all passengers desiring to exit the Car have exited. (*i.e.*, all entering passengers are blocked while there exist exiting passengers for that same floor)

1.8 If Doors become so far closed that an already-entering or already-exiting Passenger would have been prevented from entering or exiting by Door position, and Doors do not start opening within 100 msec, the passenger aborts entering/exiting the Car.

1.8.1 This abort process takes an additional 4 to 5 seconds (stochastic amount) to return to the Passenger to the position held before starting the enter/exit process.

1.8.2 This Passenger blocks all other Passengers during this process.

1.8.3 This Passenger retains queue position, and thus normally retries entry/exit immediately if possible.

1.8.4 This Passenger activates both DoorReversal[j]s.

2. Safety (environmental object)

Replication:

- One Safety object per car. This is a separate object to simplify system safety certification.

Instantiation:

- The safety system starts assuming a safe system state at initialization (initialization must ensure that an unsafe state is not transiently generated).

Input Interface:

- AtFloor[f,d]
- DoorClosed[j]
- DoorMotor[j]
- DoorReversal[j]
- HoistwayLimit[d]
- Drive
- DriveSpeed

Output Interface:

- EmergencyBrake

State:

None

BEHAVIORS:

2.1 If all AtFloor[f,stop]s are false and any DoorClosed[j] is false, set EmergencyBrake to on.

2.2 If any DoorReversal[j] is true and any DoorMotor[q] is other than open for greater than 200msec (accumulated while DoorReversal remains True), set EmergencyBrake to on.

2.3 If any HoistwayLimit[d] is True, set EmergencyBrake to on.

2.4 If a Drive command is not "adjacent to" or the same as the current DriveSpeed value for a period of longer than 100 msec, set EmergencyBrake to on.

2.4.1 The following pairs of {DriveSpeed, Drive} values are considered "adjacent":
{FastUp, SlowUp},
{SlowUp, FastUp}, {SlowUp, Stop},
{GoUp, SlowUp }, {GoUp, Stop},
{Stop, SlowUp }, {Stop, SlowDown },
{GoDown, SlowDown }, {GoDown, Stop},
{SlowDown, FastDown}, {SlowDown, Stop},
{FastDown, SlowDown}.

3. Drive (environmental object)

Replication:

- 1 Drive per Car, which tracks position of Car in hoistway as well as Drive direction/speed. The electric motor of the Drive is double-wound, so that if one winding breaks the Drive can still deliver both slow and fast speeds at approximately half the torque as for Slow and Fast of a fully operational drive. (There are many ways to deal with failure modes – this is a simple one for this project.)

Instantiation:

- Drive is Off at initialization.

Input Interface:

- Drive
- EmergencyBrake

Output Interface:

- CarPosition
- HoistwayLimit[d]
- AtFloor[f,d]

State:

- F_position[f]: an array initialized with the vertical position of each floor; used implicitly by the behaviors to determine floor position.

CONSTRAINTS:

3.1 If the EmergencyBrake is activated, it shall stop the Car regardless of Drive speed and direction.

BEHAVIORS:

3.2 Drive(Stop,d) and Drive(s,Stop) shall stop the Car regardless of values for s or d.

3.2.1 The time to Stop the Car from Fast speed depends on the Car speed before Stop is commanded and is determined by an acceleration profile.

3.2.2 The time to Stop the Car from Slow speed shall be less than 250 msec.

3.3 Drive(Slow,d), where d is not Stop, shall move the elevator at a slow speed in the appropriate direction.

3.3.1 The time to achieve Slow speed depends on speed preceding the Slow movement command and is determined by an acceleration profile.

3.3.2 The actual velocity at Slow speed depends on the drive equipment installed.

3.4 Drive(Fast,d), where d is not Stop, shall move the elevator at maximum possible speed in the appropriate direction as determined by a velocity profile.

3.5 CarPosition shall be updated according to integration of Car speed as determined by Drive() commands and an acceleration profile.

3.6 If CarPosition is greater than or equal to the position of the HoistwayLimit[Up] switch, HoistwayLimit[Up] shall be set to on and remain on.

3.7 If CarPosition is less than or equal to the position of the HoistwayLimit[Down] switch, HoistwayLimit[Down] shall be set to on and remain on.

3.8 AtFloor[f,Stop] shall be set on if and only if CarPosition is within 350 msec of travel time of Floor position f at Slow speed in either direction.

3.9 AtFloor[f,Up] shall be set on if and only if CarPosition is below the position of Floor f by a distance less than the worst-case stopping distance of the Car for that Floor in that direction.

3.10 AtFloor[f,Down] shall be set on if and only if CarPosition is above the position of Floor f by a distance less than the worst-case stopping distance of the Car for that Floor in that direction.

4. DoorMotor[j] (environmental object)

Replication:

- Each Car has two DoorMotor[j]s, with each controlled by DoorController[j]. DoorMotor[j] tracks the actual position of the door.

Instantiation:

- Both DoorMotors are off at initialization.

Input Interface:

- DoorMotor[j]

Output Interface:

- DoorClosed[j]
- DoorOpen[j]
- DoorPosition[j]

State:

- D_position[j]: float with percent open of door, range of 0 to 50.

CONSTRAINTS:

4.1 D_position[j] shall be thresholded to the range 0..50 regardless of DoorMotor[j] commands.

4.2 The Doors[j] themselves shall not activate DoorReversal[q] sensors.

4.3 DoorMotors[j] shall operate properly even if transitioned between Open and Close in either direction without an intermediate Stop command.

BEHAVIORS:

4.4 DoorMotor[j](Stop) shall stop changes in door position within 100 msec.

4.5 DoorMotor[j](Open) and DoorMotor[j](Close) shall cause door[j] to Open and Close respectively according to a velocity profile.

4.6 DoorClosed[j] shall be on if and only if DoorPosition[j] has a value less than 0.1.

4.7 DoorOpen[j] shall be on if and only if DoorPosition[j] has a value greater than 49.

4.8 DoorPosition[j] shall reflect the value of variable D_position[j].

4.9 D_position[j] shall be kept updated by a physical model to indicate current positions of simulated doors.

5. DoorControl[j]

Replication:

- Each Car has two DoorControllers[j]. Each Door[j] contributes from 0% to 50% to the DoorPosition[j] (100% = both Doors open; 50% = one Door open or both Doors half-open, or some combination; 0%= both Doors fully closed).

Instantiation:

- DoorControllers[j] shall command Doors[j] to close at initialization.

Input Interface:

- AtFloor[f,d]
- Drive
- DesiredFloor
- DesiredDwell
- DoorClosed[j]
- DoorOpen[j]
- DoorReversal[j]
- CarCall[f]
- HallCall[f,d]

Output Interface:

- DoorMotor[j]

State:

- Cycles[j], integer with number of door cycles performed; initialized to 0.
- Dwell[j], long integer with number of msec desired for door dwell during current cycle.
- CurrentFloor, is a shorthand notation for the value of whichever AtFloor[f,stop] is true, if any. If CurrentFloor is invalid it has a mnemonic value of None.
- CountDown[j]: a count-down timer for Door Dwell[j] (implemented in simulation by scheduling a future task execution at time of expiration)

CONSTRAINTS:

5.1 All DoorClosed[j] shall be true when there is no AtFloor[f,stop] that is true.

5.2 Any DoorReverse[j] cannot be true for more than an accumulated time of 50 msec without causing all DoorControllers[q] to perform an Open command.

5.3 Doors keep moving in desired direction unless commanded otherwise, subject to the constraints of the Door object.

5.4 All Doors should be commanded to identical positions at all times.

BEHAVIORS:

5.5 If AtFloor[f,d] is None set Cycles[j] to zero.

5.6 If any DoorReversal[q] is True then: command DoorMotor[j] to Open; increment Cycles[j]; set Dwell[j] to an appropriate value.

5.7 If CurrentFloor equals DesiredFloor.f, and Drive is commanded to Stop, and Cycles[j] is zero then: command DoorMotor[j] to Open; increment Cycles[j]; set Dwell[j] to an appropriate value.

5.8 If CurrentFloor equals DesiredFloor.f, and Drive is commanded to Stop, and either (HallCall[CurrentFloor,DesiredFloor.d] is true) or (any HallCall[CurrentFloor,*) is true and DesiredFloor.d is stop), then: command DoorMotor[j] to Open; increment Cycles[j]; set Dwell[j] to an appropriate value.

5.9 If CurrentFloor equals DesiredFloor.f, and Drive is commanded to Stop, and CarCall[CurrentFloor] is True, then: command DoorMotor[j] to Open; increment Cycles[j]; set Dwell[j] to an appropriate value.

5.10 When DoorOpen[j] transitions from False to True: set CountDown[j] to Dwell[j]; command DoorMotor to Stop.

5.11 When DoorClosed[j] transitions from False to True: command DoorMotor to Stop.

5.12 When CountDown[j] transitions to zero: command DoorMotor to Close.

6. DriveControl

Replication:

- There is one DriveControl, which controls the elevator Drive (the main motor moving Car Up and Down). For simplicity we will assume this node never fails, although the system could be implemented with two such nodes, one per each of the Drive windings.

Instantiation:

- DriveControl initializes to Stopping the Drive.

Input Interface:

- AtFloor[f,d]
- DoorClosed[j]
- DoorMotor[j]
- EmergencyBrake
- DesiredFloor
- DriveSpeed
- HoistwayLimit[d]

Output Interface:

- Drive

State:

- DesiredDirection = {Up, Down, Stop} computed desired direction based on comparing current floor position with Floor desired by Dispatcher. This is implicitly computed and used as a macro in the behavior descriptions.
- CurrentFloor, is a shorthand notation for the value of whichever AtFloor[f,Stop] is True, if any. If CurrentFloor is invalid it has a mnemonic value of None.

CONSTRAINTS:

6.1 Drive shall have been commanded to Stop whenever any DoorClosed is False.

6.2 Drive shall have been commanded to be Stop whenever any DoorMotor is commanded to Open.

6.3 The commanded value of Drive shall either be the same as or "adjacent to" the value of DriveSpeed.

6.3.1 The following pairs of {DriveSpeed, Drive} values are considered "adjacent":
{FastUp, SlowUp},
{SlowUp, FastUp}, {SlowUp, Stop},
{GoUp, SlowUp }, {GoUp, Stop},
{Stop, SlowUp }, {Stop, SlowDown },
{GoDown, SlowDown }, {GoDown, Stop},

{SlowDown, FastDown}, {SlowDown, Stop},
{FastDown, SlowDown}.

6.4 Drive should be Stopped whenever EmergencyBrake is activated.

BEHAVIORS:

6.5 If Drive is Stopped, and all DoorClosed[j] are True, and CurrentFloor is not equal to DesiredFloor.f, and all DoorMotor[j] are commanded to Stop, then command Drive to (Slow, DesiredDirection).

6.6 If Drivespeed is (Slow, d) and AtFloor[DesiredFloor.f,d] is False, command Drive to (Fast, d).

6.7 If Drive is commanded to (Fast, d) and AtFloor[DesiredFloor.f,d] is True, command Drive to (Slow, d).

6.8 If Drivespeed<=(Slow, d) and AtFloor[DesiredFloor.f,Stop] is True, command Drive to (Stop, Stop).

6.9 If EmergencyBrake is On, then command Drive to (Stop, Stop).

6.10 If any HoistwayLimit[d] is True, then command Drive to (Stop, Stop).

7. LanternControl[d]

Replication:

- Two controllers, one for each lantern {Up, Down} mounted in the Car by the Car Doors.

Instantiation:

- Lanterns are Off at initialization.

Input Interface:

- DoorClosed[j]
- DesiredFloor
- AtFloor[f,d]

Output Interface:

- CarLantern[d]

State:

- DesiredDirection = {Up, Down, Stop} computed desired direction based on comparing CurrentFloor with Floor desired by Dispatcher. This is implicitly computed and used as a macro in the behavior descriptions.
- CurrentFloor, is a shorthand notation for the value of whichever AtFloor[f,Stop] is True, if any. If CurrentFloor is invalid it has a mnemonic value of None.

CONSTRAINTS:

7.1 Both CarLanterns[d] shall not be On at the same time.

BEHAVIORS:

7.2 Whenever any DoorClosed[j] is False, CarLantern[DesiredDirection] shall be On.

7.2.1 If DesiredDirection is Stop, neither lantern shall illuminate.

7.3 Whenever both DoorClosed[j] are True, CarLantern[d] shall be Off.

8. HallButtonControl[f,d]

Replication:

- There are two HallButtonControllers[f,d] per floor f, one for each of the Up and Down HallCall buttons (topmost floor does not have an Up button; bottom floor does not have a Down button). These accept HallCall button presses as well as control HallLight feedback lights.

Instantiation:

- All HallCalls are false at initialization.
- All HallLights are off at initialization.

Input Interface:

- DesiredFloor
- HallCall[f,d]

Output Interface:

- HallLight[f,d]

State:
None

CONSTRAINTS:
None

BEHAVIORS:

8.1 When HallCall[f,d] is True, command HallLight[f,d] to On.

8.2 Command HallLight[DesiredFloor.f, DesiredFloor.d] to Off.

8.2.1 If DesiredFloor.d is Stop, command both HallLight[DesiredFloor.f, q] to Off.

9. CarButtonControl[f]

Replication:

- There is one CarButtonController per floor, with all controllers located in the Car. These accept CarCall button presses as well as control CarLight feedback lights.

Instantiation:

- All CarCalls are false at initialization.
- All CarLights are off at initialization.

Input Interface:

- DesiredFloor
- CarCall[f]

Output Interface:

- CarLight[f]

State:

None

CONSTRAINTS:

None

BEHAVIORS:

9.1 When CarCall[f] is True, command CarLight[f] to On.

9.2 Command CarLight[DesiredFloor.f] to Off.

10. CarPositionControl

Replication:

- There is one CarPositionControl instance in the car, which feeds values to the CarPositionIndicator.

Instantiation:

- The Car is initialized on the first Floor (Lobby).

Input Interface:

- AtFloor[f,d]
- DoorOpen[j]
- DesiredFloor

Output Interface:

- CarPositionIndicator(f)

State:

- CurrentFloor, is a shorthand notation for the value of whichever AtFloor[f,Stop] is true, if any. If CurrentFloor is invalid it has a mnemonic value of None.

CONSTRAINTS:

10.1 The Car can be at only one position at a time.

10.2 The CarPositionIndicator shall display the current floor whenever doors are open.

10.3 The floor indicated by the car position indicator shall only change by one floor in either direction per update cycle, and should be a close approximation to the car's actual position. The direction of change shall be in the same direction the Drive is moving. By "close approximation" we mean within stopping distance in the direction of motion.

BEHAVIORS:

10.4 Whenever any DoorOpen is True, CarPositionIndicator shall be commanded to display CurrentFloor.

10.5 Whenever all DoorOpens are False, CarPositionIndicator shall be commanded to display DesiredFloor.f.

11. Dispatcher

Replication:

- There is one Dispatcher in the system, corresponding with the Car.

Instantiation:

- The Dispatcher is initialized to send the car to the Lobby, have the Lobby as the desired destination, and have a preferred direction of “Stopped” (*i.e.*, no preferred direction).

Input Interface:

• AtFloor[f,d]	
• DoorClosed[j]	(optional)
• HallCall[f,d]	(optional)
• CarCall[f]	(optional)

Output Interface:

- DesiredFloor
- DesiredDwell.

State:

- Target: an integer Floor number for desired Floor, initialized to Lobby+1 = 2.
- CurrentFloor, is a shorthand notation for the value of whichever AtFloor[f,Stop] is True, if any. If CurrentFloor is invalid it has a mnemonic value of None.

CONSTRAINTS:

11.1 Target shall be a valid Floor number from 1 .. MaxFloor inclusive.

11.2 The desired direction d of DesiredFloor(f,d) shall not be Up when d = MaxFloor

11.3 The desired direction d of DesiredFloor(f,d) shall not be Down when d = 1

BEHAVIORS:

11.4 DesiredFloor.f shall always be set to Target.

11.5 DesiredFloor.d shall always be set to Stop.

11.6 Whenever any DoorClosed [j] is False, Target shall be set equal to (CurrentFloor mod MaxFloors) + 1)

11.7 DesiredDwell shall always be set to a constant appropriate value for door open dwell.