State-Charts Yakindu Assignment

By Arno Deceuninck (s0181217) and Dogukan Altay (s0211552)

Workload Distribution and Time Spent

We have divided the workload between each other and haven't done any pair programming. The distributions can be found below.

Arno Deceuninck: 6.5 hours (13 pomodoro cycles)
Normal operation (incl. tests)

Dogukan Altay: 7-8 hours
Emergency Stop feature

- Emergency Stop feature implementation (4.5 hours)
- Emergency Stop related test implementation (3-4) hours
- Report contribution

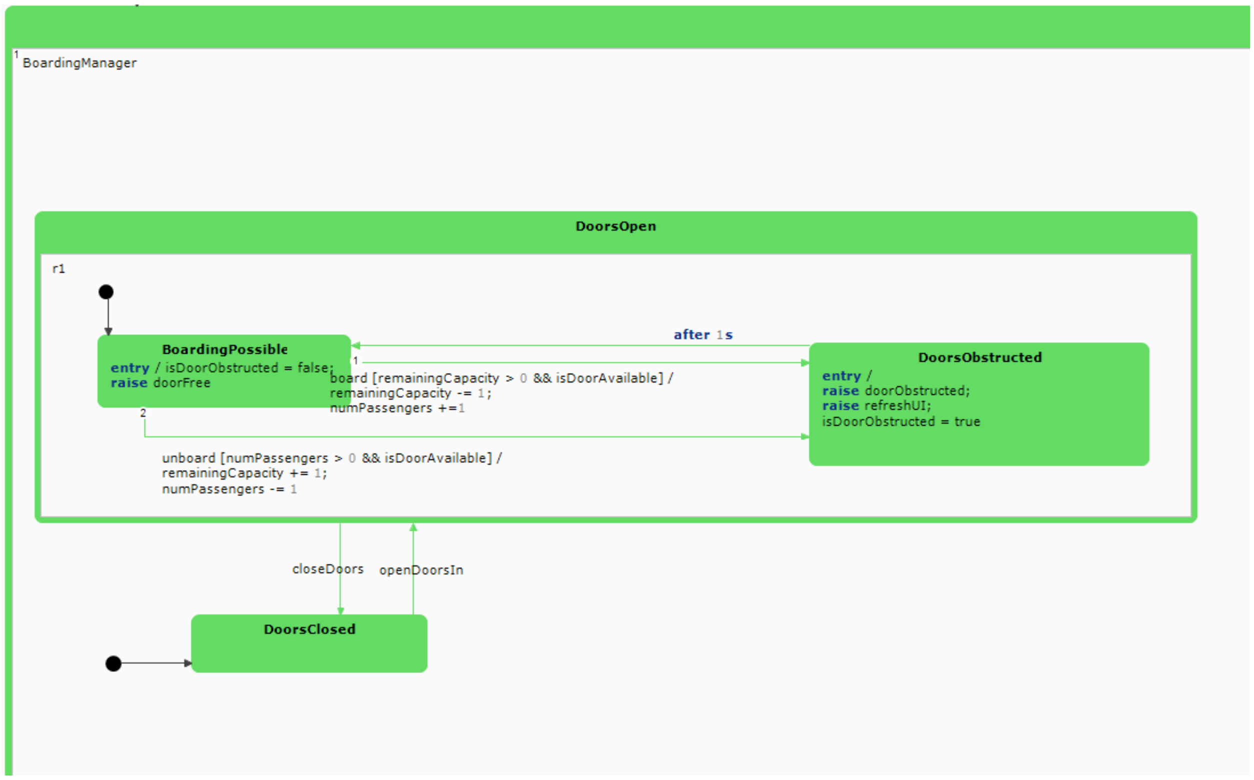
We have implemented the features in the order as shown below:

1. Arrival procedure
2. Departure procedure
3. Requesting a stop
4. Fixing 100% coverage for normal scenario
5. Small bug fixes. (Disabling (un)boarding during movement)
6. Emergency Stop feature
7. Emergency Stop feature testing

Overview of the system

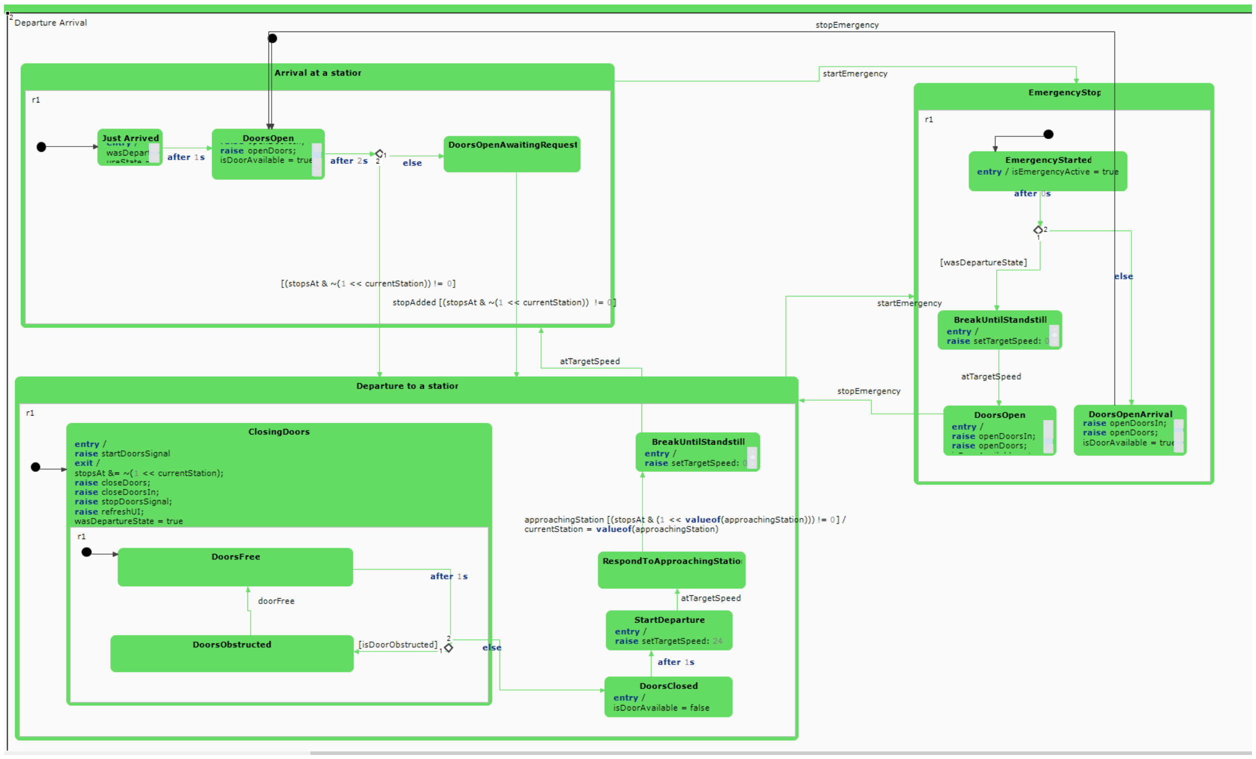
We have used a higher level of composite state called Trolley where all the features contained inside as different regions. The regions we have are:

Boarding Manager:



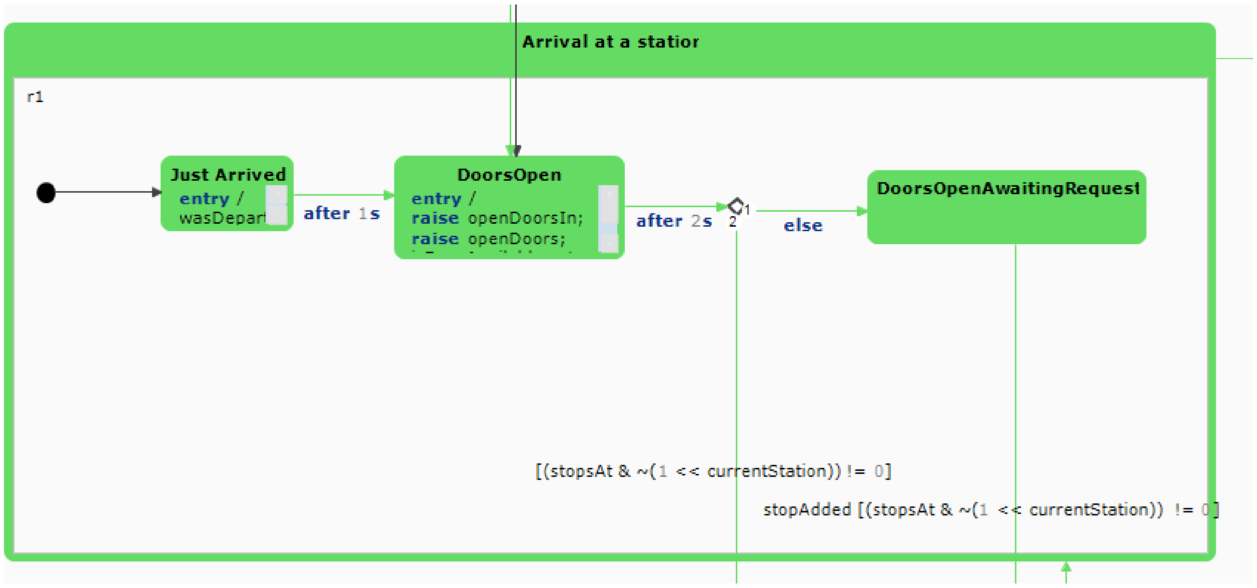
This region contains 1 composite state (DoorsOpen) and an empty state(DoorsClosed). We associated all of the (un)boarding logic within the DoorsOpen state. Basically we check all the conditions for a possible (un)boarding action and after each action we obstruct the door to meet the requirements.
Since closing doors does not involve any logic we used an empty state.

Departure Arrival



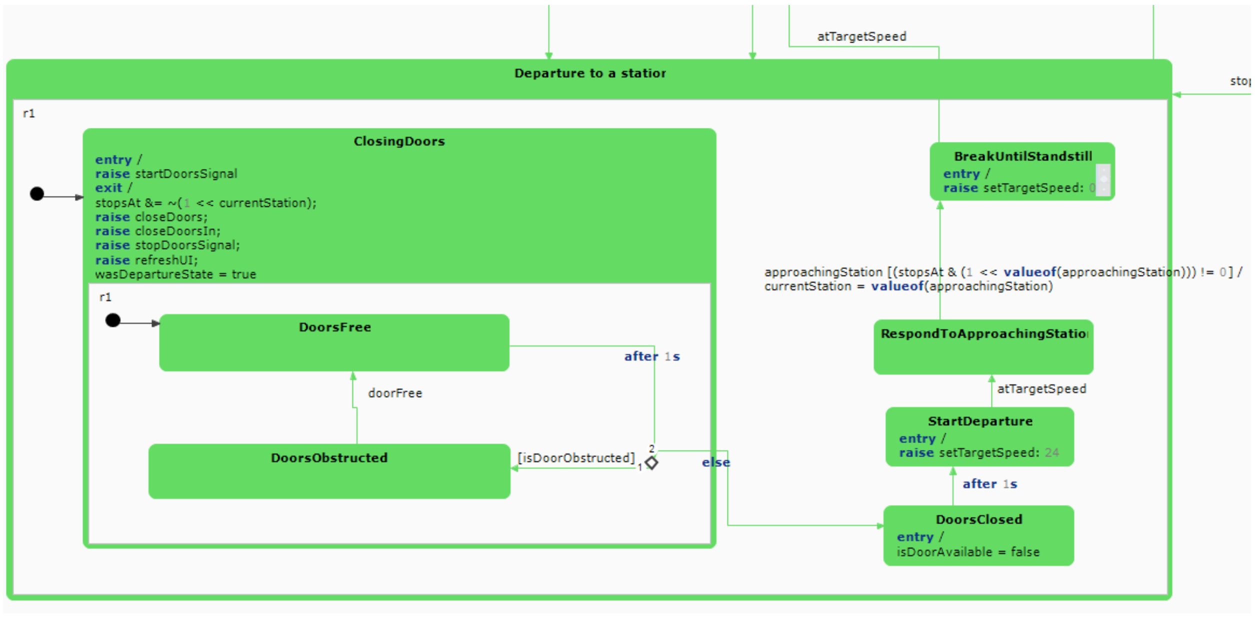
This region contains the backbones of the trolley system. We have 3 different composite states for each feature (arrival, departure and emergency stop).

a) Arrival At a Station



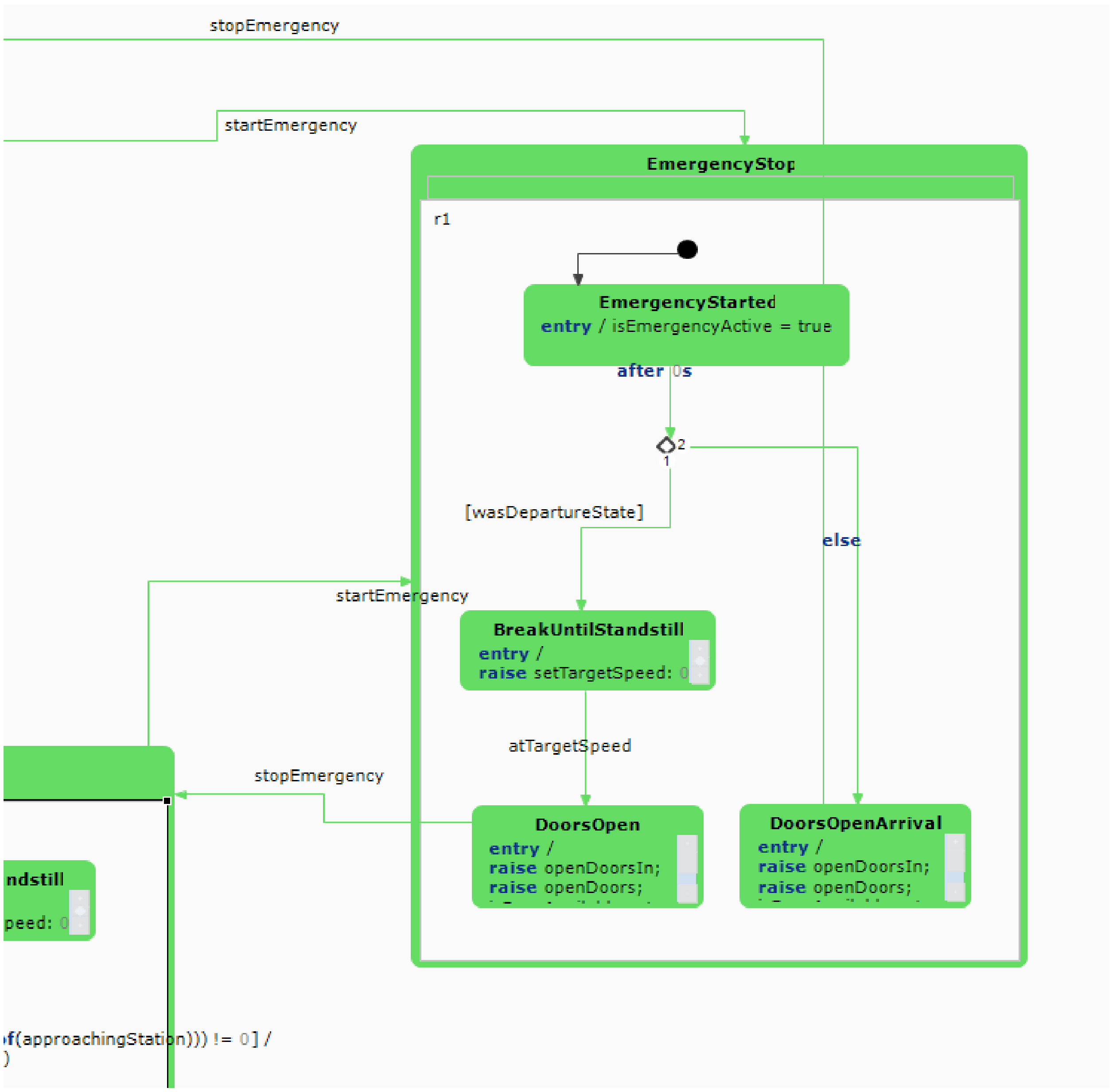
Above section represents the Arrival behaviour of the trolley. It is pretty self-explanatory overall. However the most important decision lies within the choice. System either chooses to wait while its doors open only if the stopsAt variable is equal to the currentStation the trolley is in. When a stopAdded event is triggered it again checks whether the desired stop station is not equal to the currentStation. If these conditions satisfy trolley transitions to the departure state.

b) Departure To a Station



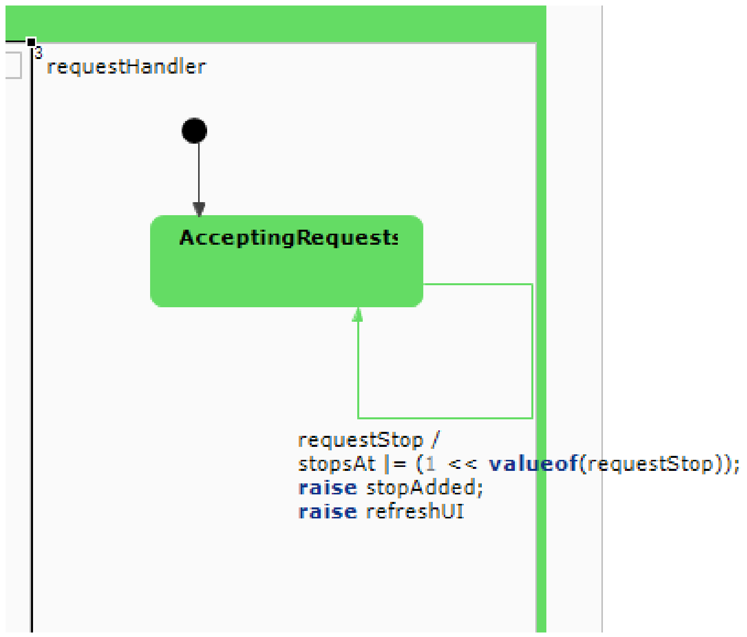
Above section represents the Departure behavior of the trolley. As soon as the current state starts at departure to a station, the system closes the doors first and right before exiting the closing door state it raises required signals. However, if the doors are obstructed due to (un)boarding it keeps the doors open until it has no action for a 1 second. After that it basically starts its departure by setting a target speed and breaks down only if the approaching station is contained in the stopsAt else it keeps target speed at 24. As soon as it stops, it transitions to the arrival state again.

c) Emergency Stop



Above section represents the Emergency Stop behavior of the trolley. It has 2 direct transitions for each state(Arrival, Departure) in the arrival departure region. StartEmergency transitions are directly connected to the top-most components state. However, due to differences in the behaviour stopEmergency transition connects to the DoorsOpen state in the arrival state since the doors are already open during an emergency stop. As seen above, the stopEmergency transitions are fired from 2 different DoorsOpen states, because we want to distinguish the last state of the trolley itself. Also stopEmergency transitions can be fired only after the doors are open.

Request Handler



```
requestStop /
stopsAt |= (1 << valueof(requestStop));
raise stopAdded;
raise refreshUI
```

This region contains the scheduling and accepting any station stop requests. Basically, it gets the requestStop input event and assigns the appropriate bit to 1 and raises refreshUI to comply with the simulation.

Test Cases

We have several test case scenarios and we have achieved 100% coverage within the Yakindu. Test cases listed below:

- TestDepartureCondition
- TestBoardPassenger
- TestBoardingWaitingPassengers
- TestBoardNonWaitingPassengers
- TestBoardAndUnboardPassengers
- TestPassengerOverflow
- TestPassengerUnderflow
- TestRushedPassenger
- TestRequestStation
- TestMultipleRequests
- TestBoarding
- TestEmergencyOpenDoorsWithoutDelayAtStation

- TestEmergencyStopStandStill
- TestStartEmergencyArrival
- TestStopEmergencyArrival
- TestEarlyStopEmergencyDeparture
- TestStopEmergencyDeparture
- TestSchedulingDuringEmergency